11-26-01

213/#7

EXPRESS MAIL CERTIFICATE

Date 11/14/01 Label No. EL767728130US

I hereby certify that, on the date indicated above, this paper or fee was deposited with the U.S. Postal Service & that it was addressed for delivery to the Assistant Commissioner for Patents, Washington, DC 20231 by "Express Mail Post Office to Addressee" service.

D B Peck

Name (Print)

Signature

PLEASE CHARGE ANY DEFICIENCY UP TO $300.00 OR CREDIT ANY EXCESS IN THE FEES DUE WITH THIS DOCUMENT TO OUR DEPOSIT ACCOUNT NO. 04-0100

Customer No.: 07278

PATENT TRADEMARK OFFICE

Docket No.: 6727/0J351US0

OIPE
NOV 14 2001
JC16
PATENT & TRADEMARK OFFICE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Itai DROR; Carmi David GRESSEL; Michael MOSTOVOY; Alexey MOLCHANOV

RECEIVED
NOV 29 2001
Technology Center 2100

Serial No.: 09/854,853 Art Unit: 2131

Confirmation No.: 8333

Filed: May 14, 2001 Examiner: N/A

For: EXTENDING THE RANGE OF COMPUTATIONAL FIELDS OF INTEGERS

CLAIM FOR PRIORITY

Hon. Commissioner of
Patents and Trademarks
Washington, DC 20231

Sir:

Applicant hereby claims priority under 35 U.S.C. Section 119 based on Israel application No. 136151 filed May 15, 2000.

A certified copy of the priority document is submitted herewith.

OIPE
NOV 14 2001
JC19
PATENT & TRADEMARK OFFICE

Respectfully submitted,

Dated: November 13, 2001

S. Peter Ludwig
Reg. No. 25,351
Attorney for Applicant(s)

DARBY & DARBY P.C.
805 Third Avenue
New York, New York 10022
212-527-7700

OIPE
NOV 14 2001
PATENT & TRADEMARK OFFICE
JC19

מדינת ישראל
STATE OF ISRAEL

RECEIVED
NOV 29 2001
Technology Center 2100

Ministry of Justice
Patent Office

משרד המשפטים
לשכת הפטנטים

This is to certify that annexed hereto is a true copy of the documents as originally deposited with the patent application of which particulars are specified on the first page of the annex.

זאת לתעודה כי רצופים בזה העתקים נכונים של המסמכים שהופקדו לכתחילה עם הבקשה לפטנט לפי הפרטים הרשומים בעמוד הראשון של הנספח.

This 17-07-2001 היום

ממונה על הבוחנים
רשם הפטנטים
Commissioner of Patents

CERTIFIED COPY OF
PRIORITY DOCUMEN

נתאשר
Certified

לשימוש הלשכה
For Office Use

מספר: Number	136151
תאריך: Date	15-05-2000
הוקדם/נדחה Ante/Post-dates	

חוק הפטנטים, התשכ"ז -- 1967
PATENTS LAW, 5727-1967

בקשה לפטנט
Application for Patent

C:38107

אני, (שם המבקש, מענו -- ולגבי גוף מאוגד -- מקום התאגדותו)
I (Name and address of applicant, and, in case of body corporate-place of incorporation)

FORTRESS U & T LTD.
P.O. Box 10072
Beer Sheva 84001
(An Israeli Company)

פורטרס א. ת. בע"מ
ת.ד. 10072
באר שבע 84001
(חברה ישראלית)

Inventors: Itai Dror
Alexey Molchanov
Michael Mostovoy
Carmi David Gressel
(Israeli citizens)

ממצאים: איתי דרור
אלכסי מולצנוב
מיכאל מוסטובוי
כרמי דוד גרסל
(אזרחים ישראלים)

בעל אמצאה מכח העברה שמה הוא By Assignment
Owner, by virtue of of an invention, the title of which is:

(בעברית) (Hebrew): שיטה והתקן להאצת מתקן מלווה מעבד הדדי קריפטוגרפי

(באנגלית) (English): METHOD AND APPARATUS FOR ACCELERATION OF CRYPTOGRAPHIC CO-PROCESSING PERIPHERALS

מבקש בזאת כי ינתן לי עליה פטנט — hereby apply for a patent to be granted to me in respect thereof

*בקשה חלוקה - Application for Division	*בקשת פטנט מוסף - Application for Patent of Addition	*דרישה דין קדימה Priority Claim		
מבקשת פטנט from Application	*לבקשה/לפטנט to Patent/Appl.	מספר.סימן Number/Mark	תאריך Date	מדינת האיגוד Convention Country
No._____.מס	No._____.מס			
dated_____יום	dated_____מיום			
רצוף בזה / עוד יוגש-*יפוי כח: כללי/מיוחד P.O.A.: general / individual - attached / to be filed later - filed in case_____הוגש בענין				
המען למסירת הודעות ומסמכים בישראל Address for Service in Israel: Sanford T. Colb & Co. P.O.B. 2273 Rehovot 76122				
חתימת המבקש Signature of Applicant: ~~For the Applicant~~, Sanford T. Colb & Co. C:38107		היום 15 This of בחודש May of שנת 2000 of the year		

לשימוש הלשכה
For Office Use

טופס זה, כשהוא מוטבע בחותם לשכת הפטנטים ומושלם בספר ובתאריך ההגשה, הינו אישור להגשת הבקשה שפרטיה רשומים לעיל.
This form, impressed with the Seal of the Patent Office and indicating the number and date of filing, certifies the filing of the application, the particulars of which are set out above.

* מחק את המיותר Delete whatever is inapplicable

שיטה והתקן להאצת מתקן מלווה מעבד הדדי קריפטוגרפי

METHOD AND APPARATUS FOR ACCELERATION OF CRYPTOGRAPHIC CO-PROCESSING PERIPHERALS

FORTRESS U & T LTD.
Inventors: Itai Dror
Alexey Molchanov
Michael Mostovoy
Carmi David Gressel

פורטרס א. ת. בע"מ
ממצאים : איתי דרור
אלכסי מולצנוב
מיכאל מוסטובוי
כרמי דוד גרסל

C: 38107

38107spc.doc

Method and Apparatus for Acceleration of Cryptographic Co-processing Peripherals

FIELD OF THE INVENTION

The present invention relates to apparatus operative to accelerate cryptographic co-processing peripherals and additionally but not exclusively to the use of such an accelerated processing apparatus for polynomial based and prime number field arithmetic, extending the range of computational fields of integers and width of serial input operands in modular arithmetic public key cryptographic co-processors designed for elliptic curve and rsa type computations

BACKGROUND OF THE INVENTION

Security enhancements and performance accelerations for computational devices are described in Applicant's U.S. Patents 5,742,530, hereinafter "P1", 5,513,133, 5,448,639, 5,261,001; and 5,206,824 and published PCT patent application PCT/IL98/00148 (WO98/50851); and U.S. Patent application 09/050958, Onyszchuk et al's U.S. Patent 4,745,568; Omura et al's U.S. Patent 4,5877,627, and applicant's U.S. Patent Application 09/480,102; the disclosures of which are hereby incorporated by reference. Applicant's U.S. Patent 5,206,824 shows an early apparatus operative to implement polynomial based multiplication and squaring, which cannot perform operations in the prime number field, and is not designed for interleaving in polynomial based computations. An additional analysis is made of an approach to use the extension field in polynomial based arithmetic in Paar, C., F. Fleischmann and P. Soria-Rodriguez, "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents", IEEE Transactions on Computers, vol. 48, No. 10, October 1999, henceforth "Paar".

SUMMARY OF THE INVENTION

It is an aim of the present invention to provide a microelectronic specialized arithmetic unit operative to perform large number computations in the polynomial based and prime integer based number fields, using similar anticipating methods for simultaneously performing interleaved modular multiplication and reduction on varied radix multipliers.

A further aim of the invention is to provide a compact microelectronic specialized arithmetic logic unit, for performing modular and normal (natural, non-negative field of integers) multiplication, division, addition, subtraction and exponentiation over very large integers. When referring to modular multiplication and squaring using both Montgomery methods and a reversed format method for simplified polynomial based multiplication and squaring, reference is made to the specific parts of the device as a superscalar modular arithmetic coprocessor, or SMAP, or MAP, also as relates to enhancements existing in the applicant's U.S. Patent pending 09/050,958 filed March 31, 1998.

According to a first aspect of the present invention there is thus provided microelectronic apparatus for performing $\otimes$ multiplication and squaring in both polynomial based GF(2^q) and GF(p) field arithmetic, squaring and reduction using a serial fed radix 2^l multiplier, B, with k character multiplicand segments, A_i, and a k character $\oplus$ accumulator wherein reduction to a limited congruence is performable "on the fly", in a systolic manner, with A_i, a multiplicand, times B, a multiplier, over a modulus, N, and a result being at most $2k + 1$ characters long, including k first emitting disregarded zero characters, which are not saved, where k characters have no less bits than the modulus, wherein said operations are carried out in two phases, the apparatus comprising;

a first (B), and second (N) main memory register, each register operative to hold at least n bit long operands, respectively operative to store a multiplier value designated B, and a modulus, denoted N, wherein the modulus is smaller than 2^n;

a digital logic sensing detector, Y_0, operative to anticipate "on the fly" when a modulus value is to be $\oplus$ added to the value in the $\oplus$ adder accumulator device such that all first k characters emitted from the device are forced to zero;

a modular multiplying device for at least k character input multiplicands, with only one, at least k characters long $\oplus$ adder, a $\oplus$ summation device operative to accept k character multiplicands, the $\otimes$ multiplication device operative to switch into the $\oplus$ accumulator device multiplicand values in turn, and in turn to receive multiplier values from a B register, and an "on the fly" simultaneously generated anticipated value as a multiplier which is operative to force k first emitting zero output characters in the

first phase, wherein at each effective machine cycle at least one designated multiplicand is $\oplus$ added into the $\oplus$ accumulation device;

the multiplicand values to be switched in turn into the $\oplus$ accumulation device consisting of one or two of the following three multiplicands, the first multiplicand being an all-zero string value, a second value, being the multiplicand A_i, and a third value, the N_0 segment of the modulus;

an anticipator to anticipate 1 bit k character serial input Y_0 multiplier values;

the apparatus being operable to input in turn multiplier values into the multiplying device in the first phase, said values being first the B operand, and concurrently, the second multiplier value consisting of the Y_0, "on the fly" anticipated k character string, to force first emitted zeroes in the output;

the apparatus further comprising an $\oplus$ accumulation device, operative to output values simultaneously as multiplicands are $\oplus$ added into the $\oplus$ accumulation device; and

an output transfer mechanism, operative in the second phase to output a final modular $\otimes$ multiplication result from the $\oplus$ accumulation device.

According to a preferred embodiment $\oplus$ summations into the $\oplus$ accumulation device are activated by each one of a series of successively newly serially loaded higher order multiplier character values.

Preferably, the multiplier characters are operative to cause no $\oplus$ summation into the $\oplus$ accumulation device if both the input B character and the corresponding input Y_0 character are zeroes;

are operative to $\oplus$ add in only the A_i multiplicand if the input B character is a one and the corresponding Y_0 character is a zero;

are operative to $\oplus$ add in only the N, modulus, if the B character is a zero, and the corresponding Y_0 character is a one; and

are operative to $\oplus$ add in the $\oplus$ summation of the modulus, N, with the multiplicand A_i if both the B input character and the corresponding Y_0 character are ones.

Preferably, the apparatus is operative to preload multiplicand values A_i and N, into two designated preload buffers, and to $\oplus$ summate these values into a third multiplicand

preload buffer, obviating the necessity of $\oplus$ adding in each multiplicand value separately.

Preferably the multiplier character values are arranged for input in serial single character form, the $\oplus$ accumulation device is arranged for output in serial single character form, and wherein the Y_0 detect device is operative to anticipate only one character in a clocked turn.

Preferably, the $\oplus$ accumulation device is operable to perform modulo 2, XOR addition/subtraction, and wherein all carry bits in addition and subtraction components are disregardable, thus not needing provisions for overflow and further limiting modular reduction in computations.

In a preferred embodiment, carry inputs are disabled to zero, and being operative to perform polynomial based multiplication.

Preferably, the apparatus is operative to provide non-carry arithmetic by setting S equal to zero acting on an element in a circuit equation computing in GF(2^q),

Preferably, the apparatus is operative to provide non-carry arithmetic by omitting carry circuitry such that the S designates omitted circuitry and reducing adders and subtractors, designated $\oplus$ to XOR, modulo 2 addition/subtraction elements.

A preferred embodiment is adapted such that the first k character segments emitted from the operational units are zeroes, zero forcing being controlled by the following four quantities in anticipating the next in turn Y_0 character:

i the l bit S_{out} bits of the result of the l bit by l bit mod 2^l $\otimes$ multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \cdot B_d$ mod 2^l;

ii the first emitting carry out character from the $\oplus$ accumulation device, S(CO_0);

iii the l bit S_{out} character from the second from the right character emitting cell of the $\oplus$ accumulation device, SO_1;

iv the l bit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register.

wherein values, $A_0 \cdot B_d$ mod 2^l, $S(CO_0)$, and SO_1 are $\oplus$ added character to character together and "on the fly" multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character to force an l bit egressing string of zeroes.

The apparatus is preferably operable to perform $\underline{\otimes}$ multiplication on polynomial based operands in a reverse mode, said multiplication comprising multiplying from right hand MS characters to left hand LS characters, said apparatus further being operative to perform modular reduced $\underline{\otimes}$ multiplication without utilizing Montgomery type parasitic functions.

Preferably, the apparatus further comprises preload buffers, which buffers are serially fed and which are connected such that multiplicand values are preloadable into the preload buffers on the fly from one or more memory devices.

The apparatus is preferably operable to $\underline{\oplus}$ sum a previously emitted value from an additional **n** bit register, S, into the output value of the $\underline{\oplus}$ accumulation device via a 1 bit $\underline{\oplus}$ adder circuit such that first emitting output characters are zeroes,

wherein the Y_0 detector is operative to detect any necessity of $\underline{\oplus}$ adding moduli to the $\underline{\oplus}$ summation in the $\underline{\oplus}$ accumulation device,

wherein the Y_0 detector is further operative to detect utilization of the next in turn $\underline{\oplus}$ added characters $A_0 \cdot B_d \bmod 2^l$, $S(CO_0)$, SO_1, S_d and $S(CO_z)$, and the composite of $\underline{\oplus}$ added characters to be finite field $\underline{\otimes}$ multiplied on the fly by the l bit J_0 value.

Preferably, for $l = 1$, J_0 is implicitly 1, and the J_0 $\underline{\otimes}$ multiplication is carried out implicitly.

Preferably, a comparator is operative to sense a finite field output from the $\underline{\otimes}$ modular multiplication device whilst operating in GF(p), wherein the first right hand emitting k zero characters are disregarded, wherein the output is larger than the modulus, N, the apparatus thereby being operative to control a modular reduction whence said value is output from a memory register to which an output stream from the multiplier device is destined, and thereby not requiring a second memory storage device for smaller ones of resulting product values.

Preferably, for $\underline{\otimes}$ modular multiplication in the GF(2^q), the apparatus is operative to carry out multiplication without an externally precomputed more than 1 bit zero-forcing factor.

A preferred embodiment is operative to compute a J_0 constant by resetting either the A operand value or the B operand value to zero and setting a partial result value, S_0, to 1.

According to a second aspect of the present invention there is provided a microelectronic apparatus for performing interleaved finite field $\otimes$ modular multiplication of two integers A and B, so as to generate an output stream of A times B modulus N, wherein a number of characters in a modulus operand register, n, is larger than a segment length of k characters wherein the $\otimes$ modular multiplication is performed in a plurality of interleaved iterations, wherein at each interleaved iteration, operands are input into a $\otimes$ multiplying device, said operands comprising any one of N, B, a previously computed partial result, S, and a k character string segment of A, the segments progressing from a first string segment A_0 to a higher string segment A_{m-1}, wherein each iterative result is $\oplus$ summated into a next temporary result, wherein at least first emitted characters of said iterations are zeroes, the apparatus comprising:

first (B), second (S) and third (N) main memory registers, each register respectively operative to store a multiplier value, a partial result value and a modulus;

a modular multiplying device operative to $\oplus$ summate into an $\oplus$ accumulation device, in turn one or two of a plurality of multiplicand values, during each one of a plurality of phases of the iterative $\otimes$ multiplication process, and in turn to receive as multipliers, inputs from:

said B register,

an "on the fly" anticipating value (Y_0) source, said anticipating value being usable as a multiplier to force first emitting right-hand zero output characters in each iteration, and

said N, register;

the multiplicand parallel registers operative at least to receive in turn, values from the A, B, and N registers, and also said zero forcing (Y_0) value;

the apparatus further comprising a zero forcing (Y_0) detect device operative to generate a binary string operative to be a multiplier during a first multiplication phase and operative to be a multiplicand in a second multiplication phase;

the apparatus being operable to obtain multiplicand values suitable for switching into the $\oplus$ accumulation device for the first multiplication phase, said values comprising firstly a zero value, secondly a value, A_i, being a k character string segment of a

multiplicand, A, and a third value N_0, being the first emitting k characters of the modulus, N;

the apparatus further being operable to utilize a temporary result value, S, resulting from a previous iteration, to be $\underline{\oplus}$ summated with a present result value emanating from the $\underline{\oplus}$ accumulation device, to generate a partial result for a next-in-turn iteration;

the apparatus further being operable to utilize multiplicand values to be input, in turn, into the $\underline{\oplus}$ accumulation device for a second multiplication phase comprising firstly a zero value, secondly an A_i operand, remaining in place from the first phase, and thirdly a Y_0 value having been anticipated in the first multiplication phase;

multiplier values input into the multiplying device in the first phase being firstly an emitted string, B_0, said multiplication device being operable to multiply said string concurrently $\underline{\otimes}$ with a second $\underline{\otimes}$ multiplier value consisting of the anticipated Y_0 string which is simultaneously loaded character by character as it is generated into a preload multiplicand buffer for the second phase;

two multiplier values operable to be input into the apparatus during a second phase being left hand n-k character values from the B operand, designated $\underline{B}$, and the left hand n-k characters of the N modulus, designated $\underline{N}$, respectively; and

wherein said apparatus further comprises a multiplying flush out device operative in a last multiplication phase to transfer a left hand segment of a result value remaining in the $\underline{\oplus}$ accumulation device into a result register.

Preferably, the apparatus is operable to perform $\underline{\otimes}$ multiplication on polynomial based operands in a reverse mode, multiplying from MS characters to LS characters, and thereby being able to perform modular reduction without Montgomery type parasitic functions.

According to a third aspect of the present invention there is provided apparatus operative in modular multiplication to anticipate a Y_0 value using first emitted values of a multiplicand, . and present inputs of a B multiplier, carry out values from a $\underline{\oplus}$ accumulation device, $\underline{\oplus}$ summation values from the $\underline{\oplus}$ accumulation device, the present values from a previously computed partial result, and carry out values from a $\underline{\oplus}$ adder which $\underline{\oplus}$ summates the result from the $\underline{\oplus}$ accumulation device with the previous partial result.

Preferably, the apparatus is adapted to ensure that k first emitted values from the device are zeroes, said adaptation comprising anticipation of a next in turn Y_0 character using the following quantities:

i l bit S_{out} bits of a result of l bit by l bit mod 2^l $\underline{\otimes}$ multiplication of the right-hand character of an A_i register times the B_d character of the B Stream, $A_0 \cdot B_d \bmod 2^l$;

ii a first emitting carry out character from the $\underline{\oplus}$ accumulation device, $S(CO_0)$;

iii the l bit S_{out} character from a second from the right-hand character emitting cell of the $\underline{\oplus}$ accumulation device, SO_1;

iv a next in turn character value from the S stream, S_d;

v a 1 bit carry out character from a Z output full adder, $S(CO_z)$;

vi a l bit J_0 value, which is a negative multiplicative inverse of a right-hand character in the N_0 modulus multiplicand register;

wherein values, $A_0 \cdot B_d \bmod 2^l$, $S(CO_0)$, SO_1, S_d are $\underline{\oplus}$ added character to character together and "on the fly" $\underline{\otimes}$ multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character.

In a further embodiment there is also provided at least one sensor operative to compare an output result to N, the mechanism operative to actuate a second subtractor on the output of the result register, thereby to output a modular reduced value which is limited congruent to the output result value, thereby avoiding any necessity to allot a second memory storage for a smaller result.

In a yet further embodiment, a value which is a $\underline{\oplus}$ summation of two multiplicands is loadable into a preload character buffer comprising at least a k character memory register whilst one of the said two multiplicands is concurrently loaded into another preload buffer.

According to a fourth aspect of the present invention there is provided apparatus with one $\underline{\oplus}$ accumulation device, and an anticipating zero forcing mechanism, operative to perform a series of interleaved $\underline{\otimes}$ modular multiplications and squarings, and being adapted to perform concurrently the equivalent of three natural integer multiplication operations, such that a result is an exponentiation.

In an embodiment, next in turn used multiplicands are preloaded into a preload register buffer on the fly.

Preferably the apparatus is operable to $\oplus$ sum two multiplicands into at least a k character register whilst concurrently loading one of the two multiplicands into a preload buffer.

In a further embodiment, apparatus buffers and registers are operative to be loaded with values from external memory sources and to be unloaded into an external memory source during computations, such that a maximum size of the operands is independent of sizes of said registers and said buffers.

In a yet further embodiment there is also provided a memory register, said memory register being typically serial –single –character -in /serial-single- character-out, parallel- at- least- k -characters- in/ parallel-at-least-k-characters-out, serial -single-character -in/ parallel –at –least –k –characters -out, and parallel- k- characters- in/ serial- single- character- out.

Preferably, the apparatus is operable to provide, during a final phase of a $\otimes$ multiplication type iteration, at inputs of said multiplication device, a plurality of zero characters, which zero characters are operative to flush out a left hand segment of a memory of the carry save $\oplus$ accumulator.

Preferably, the apparatus is operable to preload next in turn multiplicands into preload memory buffers on the fly, prior to their being required in an iteration.

Preferably, the apparatus is operable to preload multiplicand values into preload buffers on the fly from a central storage memory.

According to a fifth aspect of the present invention there is provided a microelectronic method for performing interleaved finite field $\otimes$ modular multiplication of two integers A and B, so as to generate an output stream of A times B modulus N, wherein a number of characters in a modulus operand register, n, is larger than a segment length of k characters, wherein the $\otimes$ modular multiplication is performed in a plurality of interleaved iterations, wherein at each interleaved iteration, operands are input into a $\otimes$ multiplying device, said operands comprising any one of N, B, a previously computed partial result, S, and a k character string segment of A, a multiplicand, the segments progressing from a first string segment A_0 to a higher string segment A_{m-1}, wherein each iterative result is $\oplus$ summated into a next temporary result, wherein at least first emitted characters of said iterations are zeroes, the method comprising the steps of:

$\underline{\oplus}$ summating into an $\underline{\oplus}$ accumulation device, in turn one or two of a plurality of multiplicand values, during each one of a plurality of phases of the iterative $\underline{\otimes}$ multiplication process, and in turn receiving as multipliers, inputs from:

a B register,

an "on the fly" anticipating value, Y_0, operable as a multiplier to force first emitting right-hand zero output characters in each iteration, and

an N, register:

generating a binary string operative to be a multiplier during a first multiplication phase and operative to be a multiplicand in a second multiplication phase;

obtaining multiplicand values suitable for switching into the $\underline{\oplus}$ accumulation device for the first multiplication phase consisting firstly of a zero value, secondly a value, A_i, which is a k character string segment of a multiplicand, A, and thirdly a value N_0, being the first emitted k characters of the modulus, N;

obtaining a temporary result value, S, resulting from a previous iteration, and $\underline{\oplus}$ summating said temporary result value, S, with a present result value emanating from the $\underline{\oplus}$ accumulation device, to generate a partial result for a next in turn iteration;

obtaining multiplicand values for a second multiplication phase comprising firstly a zero value, secondly an A_i operand, remaining in place from the first phase, and thirdly a Y_0 value having been anticipated in the first phase;

utilizing multiplier values obtained in the first phase, said values being firstly an emitted string, B_0, and multiplying said string concurrently $\underline{\otimes}$ with a second $\underline{\otimes}$ multiplier value consisting of the anticipated Y_0 string as it is simultaneously loaded character by character whilst being generated into a preload multiplicand buffer for the second phase;

obtaining two multiplier values during the second multiplication phase, said values being left hand n-k character values from the B operand, designated $\underline{B}$, and the left hand n-k characters of the N modulus, designated $\underline{N}$, respectively; and

in a last multiplication phase transferring a left hand segment of a result value remaining in the $\underline{\oplus}$ accumulation device into a result register.

A preferred embodiment provides the additional step of computing J_0=Y_0 for I=1 by resetting both A and B to zero and setting S_0 = 1.

Preferred embodiments of the invention described herein provide a modular computational operator for public key cryptographic applications on portable Smart Cards, typically identical in shape and size to the popular magnetic stripe credit and bank cards. Similar Smart Cards as per applicant's technology of US Patent 5,513,133 and 5,742,530, and applicants above-mentioned pending applications are being used in the new generation of public key cryptographic devices for controlling access to computers, databases, and critical installations; to regulate and secure data flow in commercial, military and domestic transactions; to decrypt scrambled pay television programs, etc. and as terminal devices for similar applications. Typically, these devices are also incorporated in computer and fax terminals, door locks, vending machines, etc.

The preferred architecture is of an apparatus operative to be integrated to a multiplicity of microcontroller and digital signal processing devices, and also reduced instruction computational designs while the apparatus operates in parallel with the host processing unit.

This embodiment preferably uses only one multiplying device which inherently serves the function of two multiplying devices, basically similar to the architecture described in applicant's US Patent No. 5,513,133 and further enhanced in U.S. Patent application 09/050,958 and PCT application PCT/IL98/0048. Using present conventional microelectronic technologies, the apparatus of the present invention may be integrated with a controlling unit with memories onto a 4 by 4.5 by 0.2 mm microelectronic circuit.

The main difference between hardware implementations in the two fields is that polynomial based additions and subtractions are simple XOR logic operations, without carry signals propagating from LS to MS. Consequently, there is no interaction between adjacent cells in the hardware implementation, and subtraction and addition are identical procedures. The earliest public notice that the authors are aware of was a short lecture by Marco Bucci of the Fondazione Ugo Bordoni, at the Eurocrypt Conference Rump Session in Perugia, Italy, in 1994.

Previous applicant's apparati were typically prepared to efficiently compute elliptic curve cryptographic protocols in the GF(p) field. In this invention we note that as there is no interaction between adjacent binary bits in the polynomial field, computations can be processed efficiently, simultaneously performing reduction and multiplication on a superscalar multiplication device. Multiplication is preferably performed as one might do with pencil and paper, starting from the most significant

partial products. Reduction is preferably performed by adding as many moduli as are necessary to reset ones to zeroes. As there is no carry out in these additions, our results are automatically modularly reduced. In this invention polynomial computations are preferably performed using the same architecture, wherein the operands may be fed in MS characters first, and wherein all internal carry signals may be forced to zero.

In a preferred embodiment of the present invention, the architecture has been extended to allow for a potentially faster progression, in that serial multipliers and results are now *l* character wide. This has somewhat complicated the anticipation process (Y_0), in that for single bit wide buses, an inversion of an odd number over a mod $2^{\wedge}$ base is also an odd number, and the least significant bit of the J_0 multiplicand was always a one. For both number fields, however the reduction process is identical, assuming *the switch out of carries*, if we remember that our only aim is to output a *k* character zero string, and we regard the Y_0 function only as a zero forcing coefficient.

The present invention also seeks to provide an architecture for a digital device which is a peripheral to a conventional digital processor, with computational, logical and architectural novel features relative to the processes described in US Patent 5,513,133.

A concurrent process and a hardware architecture are provided, to perform modular exponentiation without division preferably with the same number of operations as are typically performed with a classic multiplication/division device, wherein a classic device typically performs both a large scale multiplication and a division on each operation. A particular feature of a preferred embodiment of the present invention is the concurrency of larger scale aniticipatory zero forcing functions, the extension of number fields, and the ability to integrate this type of unit for safe communications.

The advantages realized by a preferred embodiment of this invention result from a synchronized sequence of serial processes. These processes are merged to simultaneously (in parallel) achieve three multiplication operations on *n* character operands, using one multiplexed *k* character serial/parallel multiplier in *n* effective clock cycles, where the left hand final *k* characters of the result reside in the output buffer of the multiplication device. This procedure accomplishes the equivalent of three multiplication computations in both fields, as described by Montgomery, for the prime number field.

By synchronizing loading of operands into the MAP and on the fly detecting values of operands, and on the fly preloading and simultaneous addition of next to be

used operands, the apparatus is operative to execute computations in a deterministic fashion. All multiplications and exponentiations are executed in a predetermined number of clock cycles. Additional circuitry is preferably added which, on the fly, preloads three first k character variables for a next iteration squaring sequence. A detection device is preferably provided where only two of the three operands are chosen as next iteration multiplicands, eliminating k effective clock cycle wait states. Conditional branches are replaced with local detection and compensation devices, thereby providing a basis for a simple control mechanism. The basic operations herein described may typically be executed in deterministic time using a device described in US Patent 5,513,133 to Gressel et al or devices as by STMicroelectronics in Rousset, France, under the trade name ST19-CF58.

An apparatus according to the above-described embodiments has particularly lean demands on external volatile memory for most operations, as operands are loaded into and stored in the device for the total length of the operation. The apparatus preferably exploits the CPU onto which it is appended, to execute simple loads and unloads, and sequencing of commands to the apparatus, whilst the MAP performs large number computations. Large numbers presently being implemented on smart card applications range from 128 bit to 2048 bit natural applications. The exponentiation processing time is virtually independent of the CPU which controls it. In practice, architectural changes are typically unnecessary when appending the apparatus to any CPU. The hardware device is self-contained, and is preferably appended to any CPU bus.

In general, the present invention also relates to arithmetic processing of large integers. These large numbers are typically in the natural field of (non-negative) integers or in the Galois field of prime numbers, GF(p), and also of composite prime moduli. More specifically, a preferred embodiment of the present invention seeks to provide a device that can implement modular exponentiation of large numbers. Such a device is suitable for performing the operations of Public Key Cryptographic authentication and encryption protocols, which, in the prime number field, work over increasingly large operands and which cannot be executed efficiently with present generation modular arithmetic coprocessors. Furthermore they cannot be executed securely in software implementations. The same general architecture is used in elliptic curve implementations for shorter operands, and here polynomial arithmetic may advantageously be used in order to get the right answer the first time, without the burden of the parasitic 2^{-n} factor which is discussed at length in the incorporated documents.

The architecture may offer a modular implementation of large operand integer arithmetic, while allowing for normal and smaller operand arithmetic simply by widening the serial single character bus, i.e., use of a larger radix.

For modular multiplication in the prime and composite field of odd numbers, A and B are defined as the multiplicand and the multiplier, respectively, and N is defined as the modulus in modular arithmetic. N, is typically larger than A or B. N also denotes the register where the value of the modulus is stored. N, is, in some instances, typically smaller than A. A, B, and N are typically n characters long, where characters are typically one to 8 bits long. k is the number of 1 bit characters in the size of the group defined by the size (number of cells) of the multiplying device.

In the prime field, ≡, or in some instances =, is used to denote congruence of modular numbers, for example $16 \equiv 2 \bmod 7$. 16 is termed "congruent" to 2 modulo 7 as 2 is the remainder when 16 is divided by 7. When Y mod N ≡ X mod N; both Y and X may be larger than N; however, for positive X and Y, the remainders are identical. Note also that the congruence of a negative integer Y, is Y + u N, where N is the modulus, and if the congruence of Y is to be less than N, u is the smallest integer which gives a positive result.

In $GF(2^q)$ congruence is much simpler, as addition and subtraction are identical, and the computations of embodiments of the present invention preferably never leave a substantial overflow. For $N = 1101$ and $A = 1001$, as the left hand MS bit of A is 1, we must reduce ("subtract") N from A by using modulo 2 arithmetic. A XOR N = 1001 XOR 1101 = 0100.

The Yen symbol, ¥, is used hereinbelow to denote congruence in a limited sense, especially useful in $GF(p)$. During the processes described herein, a value is often either the desired value, or equal to the desired value plus the modulus, for example in X ¥ 2 mod 7, X can be equal to 2 or 9. X is defined to have limited congruence to 2 mod 7. When the Yen symbol is used hereinbelow as a superscript, as in $B^{¥}$, then $0 \le B^{¥} < 2N$, or stated differently, $B^{¥}$ is either equal to the smallest positive B which is congruent to $B^{¥}$, or is equal to the smallest positive congruent B plus N, the modulus.

When $X = A \bmod N$, X is defined as the remainder of A divided by N; e.g., $3 = 45 \bmod 7$, and much simpler in $GF(2^q) - 1111 \bmod 1001 \equiv 0110$.

In number theory, the modular multiplicative inverse of X is written as X^{-1}, which is defined by $X\ X^{-1} \bmod N = 1$. If X = 3, and N = 13, then $X^{-1} = 9$, i.e., the remainder of 3·9 divided by 13 is 1 in GF(p).

For both number fields, we typically choose to compute the multiplicative inverse of A using the exponential function, $A^{-1} \bmod q \equiv A^{q-2} \bmod q$.

The acronyms MS and LS are used to signify "most significant" and "least significant", respectively, when referencing bits, characters, and full operand values, as is conventional in digital nomenclature, but in the reversed mode polynomial base, operands are loaded MS data first and LS last, such that the bit order of the data word is reversed when loaded. .

Throughout this specification N designates both the value N, and the name of the shift register which stores N. An asterisk superscript on a value, denotes that the value, as stands, is potentially incomplete or subject to change. A is the value of the number which is to be exponentiated, and n is the bit length of the N operand. After initialization when A is "Montgomery P field normalized" to A^* ($A^*=2^nA$ - explained in P1) A^* and N are typically constant values throughout the intermediate step in the exponentiation. In $GF(2^q)$ computations where computations might be performed with the normal unreversed positioning of bits we would be bound by this same protocol. However, using the reversed format, our computations generate most significant zeroes, which are disregarded, and do not represent a multiplication shift, as there is no carry out.

During a first iteration, after initialization of an exponentiation, B is equal to A^*. B is also the name of the register wherein the accumulated value that finally equals the desired result of exponentiation resides. S or S^* designates a temporary value; and S designates the register or registers in which all but the single MS bit of a GF(p) S is stored. (S^* concatenated with this MS bit is identical to S.) $S(i-1)$ denotes the value of S at the outset of the i'th iteration. In these polynomial computations there is no need to perform modular reduction on S.

Montgomery multiplication of X and Y in the prime number field is actually $(X{\cdot}Y{\cdot}2^{-n}) \bmod N$, where n is typically the number of characters in a modulus. This is written, $P(A{\cdot}B)_N$, and denotes MM or multiplication in the P field. In the context of Montgomery mathematics, we refer to multiplication and squaring in the P field as multiplication and squaring operations.

We may redefine this innovative extension of Montgomery arithmetic in $GF(2^q)$ to mean a reversed format data order, wherein MS zero forcing does not change

congruence, or initiate a burdensome parasitic factor. We may thus introduce a new set of symbols to accommodate the thus described arithmetic extension, and to use wider serial multiplier buses. Such an architecture is potentially vital for enabling a natural integer superscalar multiplier, which may accept 32 bit multiplicands and 4 bit multipliers, into an apparatus that can perform modular arithmetic multiplications and reductions simultaneously.

Symbols in the Serial - Parallel Super Scalar Modular Multiplier Enhancement

l Number of bits in a character (digit).

r Radix of multiplier character, $r = 2^l$.

n Size of operands (multiplier, multiplicand and modulus) in characters.

k Length of serial-parallel multiplier in characters.

m Number of interleaved slices of multiplicand. $m = (n/k)$.

S_i Result of i'th iteration; $0 \le i \le m\text{-}1$; $S_0 = 0$.

$(S_i)_0$ The right hand character of the i'th result, after disregarding the first right hand zeroes.

$\underline{S_i}$ The left hand $n\text{-}k$ characters of the i'th result.

S_{ij} j'th character of S_i.

A Parallel multiplicand consists of $m \cdot k$ characters.

A_i The i'th k character slice of A.

A_{it} The t'th character of A_i.

B Serial multiplier.

B_0 First right hand k characters of B.

$\underline{B}$ Last left hand $(n\text{-}k)$ characters of B.

B_{0j} j'th character of B_0.

$\underline{B}_j$ j'th character of $\underline{B}$.

N Modulus operand. (Often denotes both the operand and the register.)

N_0 The Right Hand k characters of N.

{ LS characters in GF(p); MS characters in GF(2^q) }

$\underline{N}$ $(n\text{-}k)$ Left Hand characters of N.

{ MS characters in GF(p); LS characters in GF(2^q) }

N_{0t} t'th character of N_0.

$\underline{N}_j$ j'th character of $\underline{N}$.

Y_0 Zero forcing variable required for both Montgomery GF(p) and GF(2^q) multiplication. Y_0 is k characters long.

Y_{0j} j'th character of Y_0.

J_{00} Zero forcing character function of the modulus, N, for "on the fly" finite field multiplication and reduction.

$Carry_j$ j'th internal carry character of radix r serial-parallel multiplier.

$Carry_a$ radix r carry of output serial adder for GF(p) computations.

Sum_j j'th internal sum character of radix r serial-parallel multiplier.

LS Least Significant.

MS Most Significant.

$\|$ Concatenation, e.g. A = 110, B = 1101 ; A $\|$ B = 1101101.

Right Hand The Least Significant portion of all GF(p) computation data blocks and the Most Significant portion of the reversed GF(2^q) format, conversely Left Hand definition.

GF(p) Galois Field, strictly speaking finite fields over prime numbers where we also use composite integers that allow for addition, subtraction, multiplication and division.

GF(2^q) Galois Fields using modulo 2 modular arithmetic.

$\underline{\oplus}$ A generic operator or device which may be switched to add or subtract integers with or without carries as befits the number system.

$\underline{\otimes}$ A generic operator or device which may be switched to multiplication over GF(p) or multiplication over GF(2^q).

S The number field switch.

S = 1, the switch is operative to enable all carry in/outputs for GF(p) computations;

S = 0, the switch is operative to disable all carry in/outputs for GF(2^q) computations.

A Serial - Parallel Super Scalar Montgomery Multiplier computes a Montgomery modular product in three phases, wherein the last phase may be a single clock dump of the left hand segment in the CSA (MS for normal multiplications, LS for reverse mode polynomial computations.)

$S_0 = 0$: partial product initially zero.

For $i = 0$ to $m - 1$: at each interleave -

The process of the first phase is the generic interaction of the operands
$(S_{i-1})_0 \underline{\oplus} A_i \cdot B_0 \underline{\oplus} Y_0 \cdot N_0$

(B_0 and Y_0 are serially character by character fed into the multiplier, A_i and N_0 are parallel operands)

The first phase process implements a $\underline{\oplus}$ summation of the two superscalar products with the right hand character of the previous result. A k zero character string is emitted from the multiplying device, and is disregarded; and a partial result is preferably left in the device buffer which is summated into the second phase result.

(B_0 and Y_0 are serially character by character fed into the multiplier,
A_i and N_0 are parallel operands).

The first phase result is R_0 concatenated with the serial out put being right hand zeroes.

The process of the second phase is the generic interaction of the operands:
$R \underline{\oplus} S_i \underline{\oplus} A_i \cdot \underline{B} \underline{\oplus} Y_0 \cdot \underline{N}$
($\underline{B}$ and $\underline{N}$ are serially fed into the multiplier, A_i and Y_0 are parallel operands)

At the end of the second phase, the left hand slice of S_i preferably remains in the Sum buffer of the Accumulator - ready to be transferred – and the right hand slice has emanated from the device, typically into an S register. Note that multiplication in the prime number field has been performed in a conventional carry save summation method. Multiplication in the GF(2^q) reversed format mode has started at the MS characters. The Y_0 function has anticipated when a modulus value must be "added" into the accumulator. Except for the disabled carry bits in the device, the process itself is identical for the two number systems.

How the Y_{0j} zero-forcing vector may be derived in finite fields.

Compute: $J_{0i} \equiv -N_{00}^{-1} \bmod r$.

This single character of the function can be hardwire implemented with random logic, with simple circuitry, or with a simple look up table. Remember there are only 2^{k-1} different values that must be derived in a look up table. In GF(p) prime

bers are odd, and the inverse of an odd number mod 2^k must always be odd. In the rse mode format, the polynomial modulus must be right justified. Therefore, if l = 1, J_0 multiplier is implicitly 1.

The result of a character output forced by the Y_0 function is always 0.

$$0 = (S_0 \oplus A_{i0} \cdot B_{0j} \oplus Y_{0j} \cdot N_{00}) \bmod r$$

Solving the above equation for Y_0, we see that J_0 is the inverse of the right hand racter of N_0.

$$Y_{0j} = (-N_{00}^{-1} \cdot (S_0 \oplus A_{i0} \cdot B_{0j})) \bmod r$$

$$Y_{0j} = (J_{0j} \cdot (S_0 \oplus A_{i0} \cdot B_{0j})) \bmod r$$

Formalizing the extended Super scalar multiplication method for both number fields:

Set $S_0 = 0$

For $i = 0$ to m-1 (Interleave iteration)

First phase:
For $j = 0$ to k-1

$$Y_{0j} = (J_{0j} \cdot (S_0 \oplus A_{i0} \cdot B_{0j})) \bmod r$$
$$S_i = (S_{i-1} \oplus A_i \cdot B_{0j} \oplus Y_{0j} \cdot N_0) / r$$

Second phase:
For $j = k$ to n-1

$$S_i = S_i \oplus A_i \cdot B_j \oplus Y_0 \cdot N_j$$

Implementation of the above algorithm with a character based serial-parallel multiplier is a simple extension of the above protocol:

Set $S_0 = 0$
For $i = 0$ to m-1 (Interleaved loop)

First phase:
For $j = 0$ to k-1

$$Y_{0j} = (J_{00} \cdot (S_{i0} \oplus A_{i0} \cdot B_{0j} \oplus S \cdot Carry_0 \oplus Sum_1 \oplus Quotient(S_{i0} \oplus Sum_0, r))) \bmod r$$

For t = 0 to k-1 (Whole loop with 1 clock pulse)

$Sum_t = (Sum_{t-1} \underline{\oplus} S \cdot Carry_t \underline{\oplus} A_{it} \cdot B_{0j} \underline{\oplus} Y_{0j} \cdot N_{0t})$ mod r

$Carry_t = (Quotient((Sum_{t-1} \underline{\oplus} Carry_t \underline{\oplus} A_{it} \cdot B_{0j} \underline{\oplus} Y_{0j} \cdot N_{0t}), r)$

(Output of multiplier in this stage is '0' s)

Second phase:

Main part

$Carry_a = 0$

For $j = k$ to n-1

For t = 0 to k-1 (Whole loop with 1 clock pulse)

$Sum_t = (Sum_{t-1} \underline{\oplus} S \cdot Carry_t \underline{\oplus} A_{it} \cdot \underline{B_j} \underline{\oplus} Y_{0t} \cdot \underline{N_j})$ mod r

$Carry_t = Quotient((Sum_{t-1} \underline{\oplus} Carry_t \underline{\oplus} A_{it} \cdot \underline{B_j} \underline{\oplus} Y_{0t} \cdot \underline{N_j}), r)$

$S_{i,j-k} = (S_{i,j-2k} \underline{\oplus} Sum_0 \underline{\oplus} S \cdot Carry_a)$ mod r

$Carry_a = Quotient((S_{i,j-2k} \underline{\oplus} Sum_0 \underline{\oplus} Carry_a), r)$

Flushing of the multiplier

For $j = n$ to $n+k$-1

For t = 0 to k-1 (Whole loop with 1 clock pulse)

$Sum_t = (Sum_{t-1} \underline{\oplus} S \cdot Carry_t)$ mod r

$Carry_t = Quotient((Sum_{t-1} \underline{\oplus} Carry_t), r)$

$S_{i,j-k} = (S_{i,j-2k} \underline{\oplus} Sum_0 \underline{\oplus} S \cdot Carry_a)$ mod r

$Carry_a = Quotient((S_{i,j-2k} \underline{\oplus} Sum_0 \underline{\oplus} Carry_a), r)$

For a formal explanation with examples of the particular case where GF(p) field, see "P1".

The above describes a microelectronic method and apparatus for p interleaved finite field $\otimes$ modular multiplication of integers A and B op generate an output stream of A times B modulus N having n characters in th operand register wherein n is larger than k, wherein the $\otimes$ multiplication performed in iterations, wherein at each interleaved iteration with operands in $\otimes$ multiplying device, the operands consisting of N, the modulus, B, a mu

terative, with progressing A_i slices. For squaring
ed from the B stream on the fly and preloaded

ion procedure, the temporary result is zero.
m previous iterations, are operative to be
om the $\oplus$ accumulation device, to generate a

in turn, into the $\oplus$ accumulation device for
nich is a pseudo register value, a second A_i
phase, and a third Y_0 value having been
ntinue multiplying the remaining characters

ltiplying device in the first phase are a first
egment of the B operand, concurrently $\otimes$
ue consisting of the anticipated Y_0 string
haracter as it is generated into a preload

apparatus during the second phase are
m the B operand, designated $\underline{B}$, and the
ated $\underline{N}$, respectively.
ice operative to transfer the left hand
mulation device. This may either be a
l. driven by zero characters fed in the

resent invention comprises a reverse
e lack of interaction between adder
ultiplication and reduction starting
a product that is the right answer,
ed zeroes which, are tantamount to
multiplication.

character string segment of A, a
A_0 string segment to the A_{m-1} string
ated into a next in turn S, temporary
of iterative results are zeroes, the

racter registers feeding the multiplying
preferably (A), configured to load the

output operands, and may respectively be
e and a modulus, N.
rably operative to $\oplus$ summate into the
plurality of multiplicand values, during
ess, and in turn to receive as multipliers,
"on the fly" anticipating value, and a Y_0
e first emitting right-hand zero output
om the modulus, N, register.
preferably operative to receive, in turn,
es, and in turn, also a multiplicand zero

erative to generate a binary string operative
eration and is operative to be a multiplicand
ication.
hed into the $\oplus$ accumulation device for the
st zero value, a second value, A_1, which is a k
d, A, and a third value N_0, being the first
The N_0 value may typically be switched in at
urth preload buffer as in Fig. 6. Then when a k
e is serially summated with the N_0 value and

ingle k character modulus, then there is no need
result value, S. If the operand is $2k$ slices or

longer, then the manipulations must be iterative, with progressing A_i slices. operations slices of B are typically snared from the B stream on the fly a into the Ai preload buffer.

At the first iteration of a multiplication procedure, the temporary resu.

Subsequent temporary results from previous iterations, are ope: $\oplus$ summated with the value emanating from the $\oplus$ accumulation device; t partial result for the next iteration in turn.

The multiplicand values to be input, in turn, into the $\oplus$ accumulatio the second phase are a first zero value, which is a pseudo register value, operand, remaining in place from the first phase, and a third Y_0 value l anticipated in the first phase operative to continue multiplying the remainin of the N modulus.

The multiplier values input into the multiplying device in the first phas emitting string, B_0, the first emitting string segment of the B operand, con multiplying with the second $\otimes$ multiplier value consisting of the anticipate which is simultaneously loaded character by character as it is generated int multiplicand buffer for the second phase.

The two multiplier values input into the apparatus during the second preferably the left hand n-k character values from the B operand, designated left hand n-k characters of the N modulus, designated $\underline{N}$, respectively.

The third phase is a flush out of the device operative to transfer the segment of a result value remaining in the $\oplus$ accumulation device. This may single clock data dump, or a simple serial unload, driven by zero characters multiplier inputs.

A particularly preferred embodiment of the present invention comprises mode multiplication in the GF(2^q). Because of the lack of interaction betw cells in this arithmetic, it is possible to perform multiplication and reductio from the MS end of the product, thereby having a product that is the righ without a burdensome parasite, caused by disregarded zeroes which, are tanta performing a right shift in conventional Montgomery multiplication.

A further preferred embodiment that allows for automatic zero forcing is an extension of the Y_0 function of patent application P1. P1 describes a device wherein only one bit is anticipated at a time. There the J_{00} bit, only, multiplies the single bit XORed values. Both the multiplicative inverse of an odd number and its negative value produce an odd number. This saves implementing a look up table or a random logic circuit to compute the J_0 value for l = 1. Note, J_0 is a different quantity in non-alike number systems. We have shown in this extension how a Y_0 value can be derived, for both relevant number fields.

The following describes the elements of the circuitry operative to anticipate the Y_0 value using first emitting values of the multiplicand, and present inputs of the B multiplier, carry out values from the $\underline{\oplus}$ accumulation device, $\underline{\oplus}$ summation values from the $\underline{\oplus}$ accumulation device, the present values from the previously computed partial result, and carry out values from the $\underline{\oplus}$ adder which $\underline{\oplus}$ summates the result from the $\underline{\oplus}$ accumulation device with the previous partial result.

Stated differently, the six values that are operative to control the zero forcing function, are:

i the l bit S_{out} bits of the result of the l bit by l bit mod 2^l $\underline{\otimes}$ multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \cdot B_d \bmod 2^l$;

ii the first emitting carry out character from the $\underline{\oplus}$ accumulation device, $S(CO_0)$:

iii the l bit S_{out} character from the second from the right-hand character emitting cell of the $\underline{\oplus}$ accumulation device, SO_1;

iv the next in turn character value from the S stream, S_d;

v the l bit carry out character from the Z output full adder, $S(CO_z)$;

vi the l bit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register:

wherein values, $A_0 \cdot B_d \bmod 2^l$, $S(CO_0)$, SO_1, S_d are $\underline{\oplus}$ added character to character together and "on the fly" $\underline{\otimes}$ multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character to force an l bit egressing character string of zeroes.

Just as in P1, in order to determine if an output must be modular reduced, a sensor operative is to compare the output result to N, the modulus, and the mechanism is itself operative to actuate a second subtractor on the output of the result register, thereby to output a modular reduced value which is limited congruent to the output result value precluding the necessity to allot a second memory storage for a smaller result.

The single $\oplus$ accumulation device, configured to perform multiplication, and an anticipating zero forcing mechanism together are operative to perform a series of interleaved $\otimes$ modular multiplications and squarings. The total device performs the equivalent of three integer multiplications, as in a conventional Montgomery method J_0 is a k character device multiplying the first k character summation of $B_0 \cdot A_i$ and S_i, and finally using the Y_0 to multiply N.

Whilst the SuperMAP computes the last iteration of a multiplication, the first slice of a next multiplication can be preloaded into a preload register buffer means on the fly. This value may be the result of a previous multiplication or a slice of a multiplicand residing in one of the register segments in the register bank of Fig. 1 or Fig. 5.

The preloaded value, which is a $\oplus$ summation of two multiplicands, is $\oplus$ summated into k slices of a register, only, for GF(2^q) computations. In GF(p) computations, provision must be made for an additional carry bit.

Especially for very long moduli, buffers and registers adjacent to the SuperMAP typically have insufficient memory resources. Means for loading operands directly into preload buffers is provided, operative to store operands in the CPU's memory map. For reverse format multiplication, bit order of input words from the CPU are typically reversed in the Data In and Data Out devices..

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the invention, and to show how the same may be carried into effect, reference will now be made, purely by way of example, to the accompanying drawings in which:

Fig. 1 is a block diagram of the apparatus according to an embodiment of the invention where four main registers are depicted, and the serial data flow path to the operational unit and the input and output data path to the host CPU of Fig. 3 are shown;

Fig. 2 is a block diagram of an embodiment of an operational unit operative to manipulate data from Fig. 1;

Fig. 3 is a simplified block diagram of a preferred embodiment of a complete single chip, monolithic cryptocomputer, typically usable in smart cards;

Fig. 4 is a simplified block diagram of a preferred embodiment of a complete single chip monolithic cryptocomputer wherein a data disable switch is operative to provide for accelerated unloading of data from the operational unit;

Fig. 5 is a simplified block diagram of a data register bank, operative to generate J_0;

Fig. 6 is a simplified block diagram of an operational unit, wherein the Y0 sensor is a device operative to force a zero first phase output;

Fig. 7A is a simplified block diagram of the main computational part of Fig. 6;

Fig. 7B is an event timing pointer diagram showing progressively the process leading to and including the first iteration of a squaring operation;

Fig. 7C is a detailed event sequence to eliminate the "Next Montgomery Squaring" delays in the first iteration of a squaring sequence.

Fig. 7D illustrates the timing of the computational output of a first iteration of a multiplication sequence, relating to Fig. 7A, 7B, and Fig. 7C; and

Fig. 8 is a simplified schematic diagram showing how to generation the Y_0 function for l = 4.

DESCRIPTION OF PREFERRED EMBODIMENTS

Figs. 1 - 2, taken together, form a simplified block diagram of a serial-parallel arithmetic logic unit (ALU) constructed and operative in accordance with a preferred embodiment of the present invention. An apparatus according to Figs. 1 - 2, preferably includes the following components:

Single Multiplexers - Controlled Switching Elements M1 to M13 select one signal or character stream from a multiplicity of inputs of signals and direct the selected signal to given outputs. The multiplexers are marked, and are intrinsic parts of larger elements.

M_K Multiplexer, 390, is an array of k+1 multiplexers, and chooses which of four k or k+1 character inputs are to be added into a CSA, 410.

In addition there are provided four registers B (1000), S_A (130), S_B (180), and N (1005), which together are the four main serial main registers in a preferred embodiment. The S_A register is conceptually and practically redundant, but can considerably accelerate very long number computations, and save volatile memory resources, especially in a case where the length of the modulus is 2·k·m characters long.

Serial-Adders and-Serial Subtractors are logic elements that have two serial character inputs and one serial character output, and summate or perform subtraction on two long strings of characters. Components 90 and 480 are subtractors, 330, and 460 are serial adders. The propagation time from input to output is very small. Serial subtractor 90 reduces B* to B if B* is larger than or equal to N. serial subtractor 480, is used as part of a comparator component to detect if B* will be larger than or equal to N. A full adder 330 adds two character streams and feeds the result into a load buffer 340. The two character streams are the same values as those stored in load buffers 290 and 320 and thus the result stored in load buffer 340 is equal to the sum of the values in load buffers 290 and 320.

The Load Buffers R1, 290; R2, 320; and R3, 340, as referred to above, are serial-in parallel-out shift registers adapted to receive the three possible more than zero multiplicand combinations.

Fast loaders and unloaders, 10 and 20, and 30 and 40, respectively, are devices to accelerate the data flow from the CPU controller. Typically these devices are DMA controlled. Devices 20 and 40 are for reversing the data word as necessary for reverse format $GF(2^q)$ multiplications.

Data In device, 50, is a parallel in serial out device, as the present ALU device is a serial fed systolic processor, and data is fed in, in parallel, and processed in serial.

Data Out device 60 is a serial in parallel out device, for outputting results from the coprocessor. A quotient generator 120 (Fig. 1) which generates a quotient character at each iteration of the dividing mechanism.

Flush Signal generators 240, 250 and 260 provide flush signals for the on Bd, S*d, and Nd registers respectively. The generators 240, 250 and 260 each preferably comprise a flush control signal Bd flush, Sd flush and Nd flush respectively anded with a data out signal. The flush signal generators are made to ensure that the last k+1

characters can flush out the CSA, as the alternative would be a complicated k+1 character parallel output element to retrieve the MS k+1 characters of the accumulator.

Latches L1, 360; L2, 370; and L3, 380; are made to receive the outputs from the load buffers 290, 320 and 340, thereby allowing the load buffers the temporal enablement to process the next phase of data before this data can be latched into L2, L2, and L3.

There is also provided a Y0 Sensor, 430, which is a logic device that determines the number of times the modulus is accumulated, in order that a k character string of LS zeros will exit at Z in ⊗ multiplications.

One character delay devices 100, 220 and 230 are inserted in the respective data streams to accommodate for synchronization problems between the data preparation devices in Fig. 1, and the data processing devices in Fig. 1.

The k character delay, shift register, 470, preferably receives the Nd result after disregarding zero output strings of the synchronized N for the larger than N comparison.

The Carry Save Accumulator 410 is almost identical to a serial/parallel multiplier, except that three different larger than zero values can be summated, instead of the single value usually latched onto the input of the s/p multiplier. When used in polynomial based computations "all carry dependent" functions are preferably disabled.

A D-type flip-flop Insert Last Carry, 440, is used to insert an mk+1'th character of the S stream as the S register is only mk characters long.

A borrow/overflow detector, 490, is connected to the output of the accumulator 410 and to the output of adder 460. It can thus either detect if the accumulator result is larger than or equal to the modulus (from N), or if the m·k·l'th bit is a one. In poly based arithmetic it would detect the first significant result bit.

The control mechanism is not depicted, but is preferably a set of cascaded counting devices, with switches set for systolic data flow.

For modular multiplication in the prime and composite prime field of numbers, we define A and B to be the multiplicand and the multiplier, and N to be the modulus which is usually larger than A or B. N also denotes the register where the value of the modulus is stored. N, may, in some instances, be smaller than A. We define A, B, and N as m·k = n character long operands. Each k character group will be called a

segment. the size of the group defined by the size of the multiplying device. Then A, B, and N are each m characters long. For ease in following the step by step procedural explanations. assume that A. B. and N are 512 bits long, (n = 512): assume that k is 64 characters long because of the present cost effective length of such a multiplier and data manipulation speeds of simple CPUs. In addition. m = 8 is the number of segments in an operand and also the number of iterations in a squaring or multiplying loop with a 512 bit operand. All operands are positive integers. More generally, A, B, N, n, k and m may assume any suitable values.

In non-modular functions, the N and S registers can be used for temporary storage of other arithmetic operands.

As discussed above, we use the symbol, ≡, to denote congruence of modular numbers. for example $16 \equiv 2 \bmod 7$, and we say 16 is congruent to 2 modulo 7 as 2 is the remainder when 16 is divided by 7. When we write $Y \bmod N \equiv X \bmod N$; both Y and X may be larger than N: however, for positive X and Y, the remainders will be identical. Note also that the congruence of a negative integer Y, is $Y + u\,N$, where N is the modulus. and if the congruence of Y is to be less than N, u will be the smallest integer which will give a positive result.

We use the symbol, ¥, to denote congruence in a more limited sense. During the processes described herein, a value is often either the desired value, or equal to the desired value plus the modulus. For example $X ¥ 2 \bmod 7$. X can be equal to 2 or 9. We say X has limited congruence to 2 mod 7.

When we write $X = A \bmod N$, we define X as the remainder of A divided by N; e.g., $3 = 45 \bmod 7$.

In number theory the modular multiplicative inverse is a basic concept. For example. the modular multiplicative inverse of X is written as X^{-1}, which is defined by $X\,X^{-1} \bmod N = 1$. If $X = 3$. and $N = 13$. then $X^{-1} = 9$. i.e., the remainder of 3·9 divided by 13 is 1.

The acronyms MS and LS are used to signify most significant and least significant when referencing bits. characters. segments. and full operand values. as is conventional in digital nomenclature.

As mentioned above, throughout this specification N designates both the value N, and the name of the shift register which contains N. An asterisk superscript on a value, denotes that the value, as stands, is potentially incomplete or subject to change. A is the value of the number which is to be exponentiated, and n is the character length of the N operand. After initialization when A is "Montgomery normalized" to A* ($A^*=2^nA$ - to be explained later) A* and N are constant values throughout the intermediate step in the exponentiation. During the first iteration, after initialization of an exponentiation, B is equal to A*. B is also the name of the register wherein the accumulated value which finally equals the desired result of exponentiation resides. S* designates a temporary value, and S, S_A and S_B designate, also, the register or registers in which all but the single MS bit of S is stored. (S* concatenated with this MS bit is identical to S.) S(i-1) denotes the value of S at the outset of the i'th iteration; S_0 denotes the LS segment of an S(i) 'th value.

We refer to the process, (defined later) $P(A\cdot B)N$ as multiplication in the P field, or sometimes, simply, a multiplication operation.

As we have used the standard structure of a serial/parallel multiplier as the basis for constructing a double acting serial parallel multiplier, we differentiate between the summating part of the multiplier, which is based on carry save accumulation, (as opposed to a carry look ahead adder, or a ripple adder, the first of which is considerably more complicated and the second very slow), and call it a carry save adder or accumulator, and deal separately with the preloading mechanism and the multiplexer and latches, which allow us to simultaneously multiply A times B and C times D, summate both results, e.g., $A\cdot B+C\cdot D$, converting this accumulator into a very powerful engine. Additional logic is added to this multiplier in order to provide for an anticipated sense operation necessary for modular reduction and serial summation necessary to provide powerful modular arithmetic and ordinary integer arithmetic on very large numbers.

Montgomery Modular Multiplication

In a classic approach for computing a modular multiplication, $A\cdot B \bmod N$, the remainder of the product $A\cdot B$ is computed by a division process. Implementing a

conventional division of large operands is more difficult to perform than serial/parallel multiplications.

Using Montgomery's modular reduction method, division is essentially replaced by multiplications using two precomputed constants. In the procedure demonstrated herein, there is only one precomputed constant, which is a function of the modulus. This constant is, or can be, computed using an ALU device according to the present invention.

A simplified presentation of the Montgomery process, as is used in a device according to a preferred embodiment of the present invention, is now provided, followed by a complete preferred description.

If we have an odd number (an LS bit one), e.g., 1010001 ($=81_{10}$) we can always transform this odd number to an even number (a single LS bit of zero) by adding to it another fixing, compensating odd number, e.g., 1111 ($=15_{10}$); as 1111 + 1010001 = 1100000 (96_{10}). In this particular case, we have found a number that produced five LS zeros, because we knew in advance the whole string, 81, and could easily determine a binary number which we could add to 81, and would produce a new binary number that would have as many LS zeros as we might need. This fixing number is be odd, else it has no effect on the progressive LS characters of a result.

If our process is a clocked serial/parallel carry save process, where it is desired to have a continuous number of LS zeros, and wherein at each clock cycle we only have to fix the next bit, at each clock it is sufficient to add the fix, if the next bit would potentially be a one or not to add the fix if the potential bit were to be a zero. However, in order not to cause interbit overflows (double carries), this fix is preferably summated previously with the multiplicand, to be added into the accumulator when the relevant multiplier bit is one, and the Y Sense also detects a one.

Now, as in modular arithmetic, we only are interested in the remainder of a value divided by the modulus, we know that we can add the modulus any number of times to a value, and still have a value that would have the same remainder. This means that we can add $YN=\Sigma y_i r^i N$ to any integer, and still have the same remainder; Y being the number of times we add in the modulus, N, to produce the required LS zeros. As described, the modulus that we add can only be odd. Methods exist wherein even

moduli are defined as r^i times the odd number that results when i is the number of LS zeros in the even number.

The problem solved by the Montgomery interleaved variations, is aimed at reducing the limited storage place we have for numbers. and the cost effective size of the multipliers. This is especially useful when performing public key cryptographic functions where we are constantly multiplying one large integer, e.g., n=1024 bit, by another large integer; a process that would ordinarily produce a double length 2048 bit integer.

We can add in Ns (the modulus) enough times to $A \cdot B=X$ or $A \cdot B+S=X$ during the process of multiplications (or squaring) so that we will have a number, Z, that has n LS zeros, and, at most, n+1 MS characters.

We can continue using such numbers, disregarding the LS n characters, if we remember that by disregarding these zeros, we have divided the desired result by r^n.

When the LS n characters are disregarded, and we only use the most significant n (or n+1) characters, then we have effectively multiplied the result by r^{-n}, the modular inverse of r^n. If we would subsequently re-multiply this result by r^n mod N (or r^n) we would obtain a value congruent to the desired result (having the same remainder) as $A \cdot B+S$ mod N. As is seen, using MM, the result is preferably multiplied by r^{2n} to overcome the r^{-n} parasitic factor reintroduced by the MM.

Example:

$A \cdot B+S \bmod N = (12 \cdot 11+10) \bmod 13 = (1100 \cdot 1011+1010)_2 \bmod 1011_2$.

$l = 1,\ r = 2$

We will add in 2^i N whenever a fix is necessary on one of the n LS bits.

	B	1011
×	A	1100
add S		1010
add A(0)·B		0000
	----	sum of LS bit = 0 not add N
add 2^0(N·0)		0000

sum 0101 →0 LS bit leaves carry save adder

add A(1)·B 0000

---- sum of LS bit = 0 - add N

add 2^1(N·1) 1101

sum 1001 →0 LS bit leaves CS adder

add A(2)·B 1011

---- sum LS bit = 0 don't add N

add 2^2(N·0) 0000

sum 1010 →0 LS bit leaves CS adder

add A(3)·B 1011

---- sum LS bit = 1 add N

add 2^3(N·1) 1101

sum 10001 →0 LS bit leaves CS adder

And the result is $10001\ 0000_2 \bmod 13 = 17 \cdot 2^4 \bmod 13$.

As 17 is larger than 13 we subtract 13, and the result is:

$17 \cdot 2^4 \equiv 4 \cdot 2^4 \bmod 13$.

formally $2^{-n}(AB+S) \bmod N = 9\ (12 \cdot 11+10) \bmod 13 \equiv 4$

In Montgomery arithmetic we utilize only the MS non-zero result (4) and effectively remember that the real result has been divided by 2^n; n zeros having been forced onto the MM result.

We have added in (8+2)·13=10·13 which effectively multiplied the result by $2^4 \bmod 13 \equiv 3$. In effect, had we used the superfluous zeros, we can say that we have performed, A·B+Y·N+S - (12·11+10·13+10) in one process, which will be described below in respect of one preferred embodiment.

Check- (12·11+10) mod 13 = 12; 4 · 3 = 12.

In summary, the result of a Montgomery Multiplication is the desired result multiplied by 2^{-n}.

To retrieve the previous result back into a desired result using the same multiplication method, we would have to Montgomery Multiply the previous result by 2^{2n}, which we will call H, as each MM leaves us with a parasitic factor of 2^{-n}.

The Montgomery Multiply function $P(A \cdot B)N$ performs a multiplication modulo N of the A·B product into the P field. (In the above example, where we derived 4). The retrieval from the P field back into the normal modular field is performed by enacting P on the result of $P(A \cdot B)N$ using the precomputed constant H. Now, if $P \equiv P(A \cdot B)N$, it follows that $P(P \cdot H)N \equiv A \cdot B \bmod N$; thereby performing a normal modular multiplication in two P field multiplications.

Montgomery modular reduction averts a series of multiplication and division operations on operands that are n and 2n characters long, by performing a series of multiplications, additions, and subtractions on operands that are n or n+1 characters long. The entire process yields a result which is smaller than or equal to N. For given A, B and odd N there is always a Q, such that $A \cdot B + Q \cdot N$ will result in a number whose n LS characters are zero, or:

$$P \cdot 2^n = A \cdot B + Q \cdot N$$

This means that we have an expression that is 2n characters long, whose n LS characters are zero.

Now, let $I \cdot r^n \equiv 1 \bmod N$ (I exists for all odd N). Multiplying both sides of the previous equation by I yields the following congruences:

from the left side of the equation:

$$P \cdot I \cdot r^n \equiv P \bmod N; \quad \text{(Remember that } I \cdot r^n \equiv 1 \bmod N)$$

and from the right side:

$$A \cdot B \cdot I + Q \cdot N \cdot I \equiv AB \cdot I \bmod N \text{ ; (Remember that } Q \cdot N \cdot I \equiv 0 \bmod N)$$

therefore:

$$P \equiv A \cdot B \cdot I \bmod N .$$

This also means that a parasitic factor $I=r^{-n} \bmod N$ is introduced each time a P field multiplication is performed.

We define the P operator such that:

$$P \equiv A \cdot B \cdot I \bmod N \equiv P(A \cdot B)N.$$

and we call this "multiplication of A times B in the P field", or Montgomery Multiplication.

The retrieval from the P field can be computed by operating P on P·H, making:

$$P(P\cdot H)N \equiv A\cdot B \bmod N ;$$

We can derive the value of H by substituting P in the previous congruence. We find:

$$P(P\cdot H)N \equiv (A\cdot B\cdot I)(H)(I) \bmod N;$$

(Note that $A\cdot B\cdot I \leftarrow P$; $H\leftarrow H$; $I\leftarrow$ and any multiplication operation introduces a parasitic I)

If H is congruent to the multiple inverse of I^2 then the congruence is valid, therefore:

$$H = I^{-2} \bmod N \equiv r^{2n} \bmod N$$

(H is a function of N and we call it the H parameter)

In conventional Montgomery methods, to enact the P operator on A·B, the following process may be employed, using the precomputed constant J:

1) $X = A\cdot B$

2) $Y = (X\cdot J) \bmod r^n$ (only the n LS characters are necessary)

3) $Z = X + Y\cdot N$

4) $S = Z / r^n$ (The requirement on J is that it forces Z to be divisible by r^n)

5) P ¥ S mod N (N is to be subtracted from S, if $S \geq N$)

Finally, at step 5) :

$$P ¥ P(A\cdot B)N,$$

[After the subtraction of N, if necessary:

$$P = P(A\cdot B)N.]$$

Following the above:

$$Y = A\cdot B\cdot J \bmod r^n$$ (using only the n LS characters) ;

and:

$$Z = A\cdot B + (A\cdot B\cdot J \bmod r^n)\cdot N.$$

In order that Z be divisible by r^n (the n LS characters of Z are preferably zero) the following congruence preferably exists:

$$[A\cdot B + (A\cdot B\cdot J \bmod r^n)\cdot N] \bmod r^n \equiv 0$$

In order that this congruence will exist, $N \cdot J \bmod r^n$ are congruent to -1 or:

$$J \equiv -N^{-1} \bmod r^n.$$

and we have found the constant J.

J, therefore, is a precomputed constant which is a function of N only. However, in a machine that outputs a MM result, character by character, provision should be made to add in Ns at each instance where the output character in the LS string would otherwise have been a zero, thereby obviating the necessity of precomputing J and subsequently computing $Y = A \cdot B \cdot J \bmod r^n$, as Y can be detected character by character using hardwired logic. As discussed in detail above, this method can only work for odd Ns.

Therefore, as is apparent, the process described employs three multiplications, one summation, and a maximum of one subtraction, for the given A, B, N, and a precomputed constant to obtain $P(A \cdot B)N$. Using this result, the same process and a precomputed constant, H, (a function of the modulo N), we are able to find A B mod N. As A can also be equal to B, this basic operator can be used as a device to square or multiply in the modular arithmetic.

Interleaved Montgomery Modular Multiplication

The previous section describes a method for modular multiplication which involved multiplications of operands which were all n characters long, and results which required 2n + 1 characters of storage space.

Using Montgomery's interleaved reduction (as described in the aforementioned paper by Dussé), it is possible to perform the multiplication operations with shorter operands, registers, and hardware multipliers; enabling the implementation of an electronic device with relatively few logic gates.

First we will describe how the device can work, if at each iteration of the interleave, we compute the number of times that N is added, using the J_0 constant. Later, we describe how to interleave, using a hardwire derivation of Y_0, which will eliminate the J_0- phase of each multiplication {2) in the following example}, and enable us to integrate the functions of two separate serial/multipliers into the new single generic

multiplier which can perform A·B+C·N+S at better than double speed using similar silicon resources.

Using a k character multiplier, it is convenient to define segments of k character length: there are m segments in n characters: i.e., m·k = n.

J_0 will be the LS segment of J.

Therefore:

$J_0 \equiv -N_0^{-1} \bmod r^k$ (J_0 exists as N is odd).

Note, the J and J_0 constants are compensating numbers that when enacted on the potential output, tell us how many times to add the modulus, in order to have a predefined number of least significant zeros. We will later describe an additional advantage to the present serial device; since, as the next serial bit of output can be easily determined, we can always add the modulus (always odd) to the next intermediate result. This is the case if, without this addition, the output character, the LS serial bit exiting the CSA, would have been a "1"; thereby adding in the modulus to the previous even intermediate result, and thereby promising another LS zero in the output string. Remember, congruency is maintained, as no matter how many times the modulus is added to the result, the remainder is constant.

In the conventional use of Montgomery's interleaved reduction, $P(A \cdot B)N$ is enacted in m iterations as described in the following steps (1) to (5):

Initially S(0) = 0 (the ¥ value of S at the outset of the first iteration).

For i = 1, 2....m :

1) $X = S(i-1) + A_{i-1} \cdot B$ (A_{i-1} is the i-1 th character of A ; S(i-1) is the value of S at the outset of the i'th iteration.)

2) $Y_0 = X_0 \cdot J_0 \bmod r^k$ (The LS k characters of the product of $X_0 \cdot J_0$)

(The process uses and computes the k LS characters only, e.g., the least significant 64 characters)

In the preferred implementation, this step is obviated, because in a serial machine Y_0 can be anticipated character by character.

3) $Z = X + Y_0 \cdot N$

4) $S(i) = Z/r^k$ (The k LS characters of Z are always 0, therefore Z is always divisible by r^k. This division is tantamount to a k character right shift as the LS k

characters of Z are all zeros; or as will be seen in the circuit, the LS k characters of Z are simply disregarded.

(5) $S(i) = S(i) \bmod N$ (N is to be subtracted from those S(i)'s which are larger than N).

Finally, at the last iteration (after the subtraction of N, when necessary), $C = S(m) = P(A \cdot B)N$.

To derive $F = A \cdot B \bmod N$, the P field computation, $P(C \cdot H)N$, is performed

It is desired to know, in a preferred embodiment, that for all S(i)'s, S(i) is smaller than 2N. This also means, that the last result (S(m)) can always be reduced to a quantity less than N with, at most, one subtraction of N.

We observe that for operands which are used in the process:

$S(i-1) < r^{n+1}$ (the temporary register can be one bit longer than the B or N register),

$B < N < r^n$ and $A_{i-1} < r^k$.

By definition:

$S(i) = Z/r^k$ (The value of S at the end of the process, before a possible subtraction)

For all Z, $Z(i) < r^{n+k-1}$.

$$X_{max} = S_{max} + A_i \cdot B < r^{n+1} - 1 + (r^k-1)(r^n-1)$$

$$Q_{max} = Y_0 N < (r^k-1)(r^n-1)$$

therefore:

$$Z_{max} < r^{k+n-1} - r^{k+1} + 1 < r^{k+n+1} - 1.$$

and as Z_{max} is divided by r^k:

$$S(m) < r^{n+1} - r^1.$$

Because $N_{min} > r^n - r$, $S(m)_{max}$ is always less than $2 \cdot N_{min}$, one subtraction is all that is necessary on a final result.

$$S(m)_{max} - N_{min} = (r^{n+1} - r^1 - 1) - (r^n - 1) = r^n - 4 < N_{min}.$$

Example of a Montgomery interleaved modular multiplication:

The following computations in the hexadecimal format clarify the meaning of the interleaved method:

N = a59, (the modulo), A = 99b, (the multiplier), B = 5c3 (the multiplicand), n = 12, r = 2, (the character length of N), k = 4, (the size in characters of the multiplier and also the size of a segment), and m = 3, as n = k·m.

J_0 = 7 as 7·9 ≡ -1 mod 16 and H ≡ $2^{2 \cdot 12}$ mod a59 ≡ 44b.

The expected result is F ≡ A·B mod N ≡ 99b·5c3 mod a59 ≡ 375811 mod a59 = 220_{16}.

Initially: S(0) = 0

Step 1

X = S(0) + A_0·B = 0 + b·5c3 = 3f61

Y_0 = X_0· J_0 mod r^k = 7 (Y_0 - hardwire anticipated in new MAP)

Z = X + Y_0·N = 3f61 + 7·a59 = 87d0

S(1) = Z / r^k = 87d

Step 2

X = S(1) + A_1·B = 87d + 9·5c3 = 3c58

Y_0 = X_0· J_0 mod r^k = 8·7 mod 2^4 = 8 (Hardwire anticipated)

Z = X + Y_0·N = 3c58 + 52c8 = 8f20

S(2) = Z / r^k = 8f2

Step 3

X = S(2) + A_2·B = 8f2 + 9·5c3 = 3ccd

Y_0 = d·7 mod 2^4 = b (Hardwire anticipated)

Z = X + Y_0·N = 3ccd + b·a59 = aea0

S(3) = Z / r^k = aea ,

as S(3) > N ,

S(m)=S(3) - N = aea - a59 = 91

Therefore C = P(A·B)N = 91_{16}.

Retrieval from the P field is performed by computing P(C·H)N:

Again initially: S(0) = 0

Step 1

X = S(0) + C_0·H = 0 + 1·44b = 44b

Y_0 = d (Hardwire anticipated in new MAP)

$Z = X + Y_0 \cdot N$ = 44b + 8685 = 8ad0

$S(1) = Z / r^k$ = 8ad

Step 2

$X = S(1) + C_1 \cdot H$ = 8ad + 9·44b = 2f50

Y_0 = 0 (Hardwire anticipated in new MAP)

$Z = X + Y_0 \cdot N$ = 2f50 + 0 = 2f50

$S(2) = Z / r^k$ = 2f5

Step 3

$X = S(2) + C_2 \cdot H$ = 2f5 + 0·44b = 2f5

Y_0 = 3 (Hardwire anticipated in new MAP)

$Z = X + Y_0 \cdot N$ = 2f5 + 3·a59 = 2200

$S(3) = Z / r^k = 220_{16}$

which is the expected value of 99b 5c3 mod a59.

If at each step we disregard k LS zeros, we are in essence multiplying the n MS characters by r^k. Likewise, at each step, the i'th segment of the multiplier is also a number multiplied by r^{ik}, giving it the same rank as S(i).

It may also be noted that in another preferred embodiment, it is of potential value to know the J_0 constant.

Exponentiation

The following derivation of a sequence [D. Knuth, *The art of computer programming*, vol. 2: Seminumerical algorithms, Addison-Wesley, Reading Mass., 1981] hereinafter referred to as "Knuth", explains a sequence of squares and multiplies, which implements a modular exponentiation.

After precomputing the Montgomery constant, $H= 2^{2n}$, as this device can both square and multiply in the P field, we compute:

$C = A^E \bmod N$.

Let E(j) denote the j bit in the binary representation of the exponent E, starting with the MS bit whose index is 1 and concluding with the LS bit whose index is q, we can exponentiate as follows for odd exponents:

A^* ¥ P(A·H)N A* is now equal to $A \cdot 2^n$.

B = A^*

FOR j = 2 TO q-1

B ¥ P(B·B)N

IF E(j) = 1 THEN

B ¥ P(B·A^*)N

ENDFOR

B ¥ P(B·A)N E(0)=1; B is the last desired temporary result multiplied by 2^n.

A is the original A.

C = B

C= C-N if C≥N.

After the last iteration, the value B is ¥ to A^E mod N, and C is the final value.

To clarify, we shall use the following example:

E = 1011 ⟶ E(1) = 1; E(2) = 0; E(3) = 1; E(4) = 1;

To find A^{1011} mod N; q = 4

A^* = P(A·H)N = AI^{-2} I=AI^{-1} mod N

B = A^*

FOR j = 2 to q

B = P(B·B)N which produces: $A^2(I^{-1})^2 \cdot I = A^2 \cdot I^{-1}$

E(2) = 0: B = $A^2 \cdot I^{-1}$

j = 3 B = P(B·B)N = $A^2(I^{-1})^2 \cdot I = A^4 \cdot I^{-1}$

$E(3) = 1$ $B = P(B \cdot A^*)N = (A^4 \cdot I^{-1})\,(AI^{-1}) \cdot I = A^5 \cdot I^{-1}$

$j = 4$ $\quad B = P(B \cdot B)N = A^{10} \cdot I^{-2} \cdot I = A^{10} \cdot I^{-1}$

As E(4) was odd, the last multiplication will be by A, to remove the parasitic I^{-1}.

$B = P(B \cdot A)N = A^{10} \cdot I^{-1} \cdot A \cdot I = A^{11}$

$C = B$

A method for computing the H parameter by a reciprocal process is described in US Patent 5,513,133.

Reference is now made to Fig. 3 which is a simplified block diagram showing how the present invention may be implemented in a smart card. An internal bus 500 links components including a CPU 502, a RAM 504, a ROM 506, a controlled access ROM 508, and modular arithmetic co-processor 510. As shown herein, the co-processor 510 is connected via data 512 and control 514 registers to the internal bus 500. The controlled access ROM 508 is connected via address and data latch means 516 and a control and test register 518. Various other devices may be attached to the bus such as a physical sequence random generator 520, security logic 522 and interfacing and resetting circuitry 524 and 526 respectively.

When a cryptographic program such as verifying an RSA signature is run, it may require modular arithmetic functions such as modular exponentiation. The cryptographic program that calls the cryptographic function is preferably run on the CPU 502. However the modular arithmetic function is carried out on the co-processor 510, the CPU 502 serving only to assist with storage and like menial tasks.

Reference is now made to Fig. 4 which is another simplified block diagram of an implementation of the present invention for use in a smart card. Parts that are the same as those shown in Fig. 3 are given the same reference numerals and are not described again, except as necessary for an understanding of the present embodiment. In Fig. 4 the CPU 502 is shown with an external accumulator 530. Detaching the CPU Accumulator from the Data Bus 500, while unloading data from the arithmetic co-processor enables direct transfer of data from the SMAP to memory.

Fig. 5 is simplified block diagram of a preferred embodiment of a data register bank within a co-processor such as co-processor 510, with a J_0 generator. The

co-processor 510 is connected to a data bus with a CPU as in previous figures. Parts that are the same as those shown in previous figures are given the same reference numerals and are not described again, except as necessary for an understanding of the present embodiment. A register bank 540 comprises a B register 542, an A register 544, an S register 546, and an N register 548. The outputs of each of the registers are connected to a serial data switch and serial process_conditioner 550 which in turn is connected to an operational unit 552 which carries out the modular arithmetic operations. Connected between the N register 548 and the operational unit 552 is a J_0 generator 552.

In the embodiment the J_0 generator compiles a 1 bit primary zero forcing function for use in the modular arithmetic functions described above.

Fig. 6 is a simplified internal block diagram of the operational unit of Fig. 5. The unit preferably supports accelerated squaring operations, in that the additional Y0B0 serial buffer accepts a Y0 value in the first multiplication phase, and in the second multiplication phase, a modular reduced B_0 is used for

Reference is now made to Fig. 7a which is a block diagram of the main computational part of the operational unit of Fig. 6. Numbers appearing in circles relate to the sequence diagrams of Figs. 7B to 7D.

Reference is now made to Fig. 7b which is an event timer pointer diagram showing progressively the process leading to and including the first iteration of a squaring operation.

Reference is now made to Fig. 7c which is a generalized event sequence showing a method of eliminating the Next Montgomery Squaring delays in a first iteration of a squaring sequence. Circled numbers refer to Figs. 7a, b and d.

Reference is now made to Fig. 7d which is a generalized event timer pointer diagram illustrating the timing of the computational output of the first iteration of a squaring operation.

Reference is now made to Fig. 8 which is a schematic diagram for designing a 4 bit Y_0 zero forcing function. The variable inputs into the force function f_2 are the N_0 bits, the four S_0 bits, and the four multiplier and multiplicand bits, A_{i0}, B_{0i}, and the carry

switch. S. f_1 is random logic, the A and B bits are input into a $\otimes$ multiplier and $\oplus$ added to the S_0 partial result. When S = 0, all carries are disconnected.

It is appreciated that various features of the invention which are, for clarity, described in the contexts of separate embodiments may also be provided in combination in a single embodiment. Conversely, various features of the invention which are, for brevity, described in the context of a single embodiment may also be provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in the art that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention includes both combinations and subcombinations of the various features described hereinabove as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description and which are not in the prior art.

In the following claims, symbols such as $\oplus$ and $\otimes$ have the meanings given in the preceding description.

CLAIMS

1. Microelectronic apparatus for performing $\otimes$ multiplication and squaring in both polynomial based GF(2^q) and GF(p) field arithmetic, squaring and reduction using a serial fed radix 2^l multiplier, B, with k character multiplicand segments, A_i, and a k character $\oplus$ accumulator wherein reduction to a limited congruence is performable "on the fly", in a systolic manner, with A_i, a multiplicand, times B, a multiplier, over a modulus, N, and a result being at most $2k + 1$ characters long, including k first emitting disregarded zero characters, which are not saved, where k characters have no less bits than the modulus, wherein said operations are carried out in two phases, the apparatus comprising:

a first (B), and second (N) main memory register, each register operative to hold at least **n** bit long operands, respectively operative to store a multiplier value designated B, and a modulus, denoted N, wherein the modulus is smaller than 2^n;

a digital logic sensing detector, Y_0, operative to anticipate "on the fly" when a modulus value is to be $\oplus$ added to the value in the $\oplus$ adder accumulator device such that all first k characters emitted from the device are forced to zero;

a modular multiplying device for at least k character input multiplicands, with only one, at least k characters long $\oplus$ adder, a $\oplus$ summation device operative to accept k character multiplicands, the $\otimes$ multiplication device operative to switch into the $\oplus$ accumulator device multiplicand values in turn, and in turn to receive multiplier values from a B register, and an "on the fly" simultaneously generated anticipated value as a multiplier which is operative to force k first emitting zero output characters in the first phase, wherein at each effective machine cycle at least one designated multiplicand is $\oplus$ added into the $\oplus$ accumulation device;

the multiplicand values to be switched in turn into the $\oplus$ accumulation device consisting of one or two of the following three multiplicands, the first multiplicand being an all-zero string value, a second value, being the multiplicand A_i, and a third value, the N_0 segment of the modulus;

an anticipator to anticipate 1 bit k character serial input Y_0 multiplier values;

the apparatus being operable to input in turn multiplier values into the multiplying device in the first phase, said values being first the B operand, and concurrently, the second multiplier value consisting of the Y_0, "on the fly" anticipated k character string, to force first emitted zeroes in the output;

the apparatus further comprising an $\oplus$ accumulation device, operative to output values simultaneously as multiplicands are $\oplus$ added into the $\oplus$ accumulation device; and

an output transfer mechanism, operative in the second phase to output a final modular $\otimes$ multiplication result from the $\oplus$ accumulation device.

2. Apparatus as in claim 1 wherein $\oplus$ summations into the $\oplus$ accumulation device are activated by each one of a series of successively newly serially loaded higher order multiplier character values.

3. Apparatus as in claim 1 or claim 2, wherein the multiplier characters are operative to cause no $\oplus$ summation into the $\oplus$ accumulation device if both the input B character and the corresponding input Y_0 character are zeroes;

are operative to $\oplus$ add in only the A_i multiplicand if the input B character is a one and the corresponding Y_0 character is a zero;

are operative to $\oplus$ add in only the N, modulus, if the B character is a zero, and the corresponding Y_0 character is a one; and

are operative to $\oplus$ add in the $\oplus$ summation of the modulus, N, with the multiplicand A_i if both the B input character and the corresponding Y_0 character are ones.

4. Apparatus as in claim 1, 2 or 3, operative to preload multiplicand values A_i and N, into two designated preload buffers, and to $\oplus$ summate these values into a third multiplicand preload buffer, obviating the necessity of $\oplus$ adding in each multiplicand value separately.

5. Apparatus according to any preceding claim, wherein the multiplier character values are arranged for input in serial single character form, the

$\underline{\oplus}$ accumulation device is arranged for output in serial single character form, and wherein the Y_0 detect device is operative to anticipate only one character in a clocked turn.

6. Apparatus according to any preceding claim, wherein the $\underline{\oplus}$ accumulation device is operable to perform modulo 2, XOR addition/subtraction, and wherein all carry bits in addition and subtraction components are disregardable, thus not needing provisions for overflow and further limiting modular reduction in computations.

7. Apparatus according to any preceding claim wherein carry inputs are disabled to zero, and being operative to perform polynomial based multiplication.

8. Apparatus according to any preceding claim, operative to provide non-carry arithmetic by setting S equal to zero acting on an element in a circuit equation computing in GF(2^q),

9. Apparatus according to any preceding claim operative to provide non-carry arithmetic by omitting carry circuitry such that the S designates omitted circuitry and reducing adders and subtractors, designated $\underline{\oplus}$ to XOR, modulo 2 addition/subtraction elements.

10. Apparatus as in claim 1 adapted such that the first k character segments emitted from the operational units are zeroes, zero forcing being controlled by the following four quantities in anticipating the next in turn Y_0 character:

i the l bit S_{out} bits of the result of the l bit by l bit mod 2^l $\underline{\otimes}$ multiplication of the right-hand character of the A_i register times the B_d character of the B Stream, $A_0 \cdot B_d$ mod 2^l;

ii the first emitting carry out character from the $\underline{\oplus}$ accumulation device, S(CO_0);

iii the l bit S_{out} character from the second from the right character emitting cell of the $\underline{\oplus}$ accumulation device, SO_1;

iv the l bit J_0 value, which is the negative multiplicative inverse of the right-hand character in the N_0 modulus multiplicand register.

wherein values, $A_0 \cdot B_d \bmod 2^l$, $S(CO_0)$, and SO_1 are $\oplus$ added character to character together and "on the fly" multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character to force an l bit egressing string of zeroes.

11. Apparatus according to any preceding claim, operable to perform $\otimes$ multiplication on polynomial based operands in a reverse mode, said multiplication comprising multiplying from right hand MS characters to left hand LS characters, said apparatus further being operative to perform modular reduced $\otimes$ multiplication without utilizing Montgomery type parasitic functions.

12. Apparatus according to any preceding claim, further comprising preload buffers, which buffers are serially fed and which are connected such that multiplicand values are preloadable into the preload buffers on the fly from one or more memory devices.

13. Apparatus according to any preceding claim, operable to $\oplus$ sum a previously emitted value from an additional **n** bit register, S, into the output value of the $\oplus$ accumulation device via a 1 bit $\oplus$ adder circuit such that first emitting output characters are zeroes.

wherein the Y_0 detector is operative to detect any necessity of $\oplus$ adding moduli to the $\oplus$ summation in the $\oplus$ accumulation device.

wherein the Y_0 detector is further operative to detect utilization of the next in turn $\oplus$ added characters $A_0 \cdot B_d \bmod 2^l$, $S(CO_0)$, SO_1, S_d and $S(CO_z)$, and the composite of $\oplus$ added characters to be finite field $\otimes$ multiplied on the fly by the l bit J_0 value.

14. Apparatus according to claim 10 or claim 13, wherein for $l = 1$, J_0 is implicitly 1, and the J_0 $\otimes$ multiplication is carried out implicitly.

15. Apparatus according to any preceding claim, wherein a comparator is operative to sense a finite field output from the $\otimes$ modular multiplication device whilst

operating in GF(p), wherein the first right hand emitting k zero characters are disregarded, wherein the output is larger than the modulus, N, the apparatus thereby being operative to control a modular reduction whence said value is output from a memory register to which an output stream from the multiplier device is destined, and thereby not requiring a second memory storage device for smaller ones of resulting product values.

16. Apparatus according to any preceding claim, which, for $\otimes$ modular multiplication in the GF(2^q), is operative to carry out multiplication without an externally precomputed more than 1 bit zero-forcing factor.

17. Apparatus according to any preceding claim, operative to compute a J_0 constant by resetting either the A operand value or the B operand value to zero and setting a partial result value, S_0, to 1.

18. Microelectronic apparatus for performing interleaved finite field $\otimes$ modular multiplication of two integers A and B, so as to generate an output stream of A times B modulus N, wherein a number of characters in a modulus operand register, n, is larger than a segment length of k characters wherein the $\otimes$ modular multiplication is performed in a plurality of interleaved iterations, wherein at each interleaved iteration, operands are input into a $\otimes$ multiplying device, said operands comprising any one of N, B, a previously computed partial result, S, and a k character string segment of A, the segments progressing from a first string segment A_0 to a higher string segment A_{m-1}, wherein each iterative result is $\oplus$ summated into a next temporary result, wherein at least first emitted characters of said iterations are zeroes, the apparatus comprising:

first (B), second (S) and third (N) main memory registers, each register respectively operative to store a multiplier value, a partial result value and a modulus;

a modular multiplying device operative to $\oplus$ summate into an $\oplus$ accumulation device, in turn one or two of a plurality of multiplicand values, during each one of a plurality of phases of the iterative $\otimes$ multiplication process, and in turn to receive as multipliers, inputs from:

said B register.

an "on the fly" anticipating value (Y_0) source, said anticipating value being usable as a multiplier to force first emitting right-hand zero output characters in each iteration, and

said N. register;

the multiplicand parallel registers operative at least to receive in turn, values from the A. B. and N registers, and also said zero forcing (Y_0) value;

the apparatus further comprising a zero forcing (Y_0) detect device operative to generate a binary string operative to be a multiplier during a first multiplication phase and operative to be a multiplicand in a second multiplication phase;

the apparatus being operable to obtain multiplicand values suitable for switching into the $\underline{\oplus}$ accumulation device for the first multiplication phase, said values comprising firstly a zero value, secondly a value, A_i, being a k character string segment of a multiplicand, A, and a third value N_0, being the first emitting k characters of the modulus, N;

the apparatus further being operable to utilize a temporary result value, S, resulting from a previous iteration, to be $\underline{\oplus}$ summated with a present result value emanating from the $\underline{\oplus}$ accumulation device, to generate a partial result for a next-in-turn iteration;

the apparatus further being operable to utilize multiplicand values to be input, in turn, into the $\underline{\oplus}$ accumulation device for a second multiplication phase comprising firstly a zero value, secondly an A_i operand, remaining in place from the first phase, and thirdly a Y_0 value having been anticipated in the first multiplication phase;

multiplier values input into the multiplying device in the first phase being firstly an emitted string, B_0, said multiplication device being operable to multiply said string concurrently $\underline{\otimes}$ with a second $\underline{\otimes}$ multiplier value consisting of the anticipated Y_0 string which is simultaneously loaded character by character as it is generated into a preload multiplicand buffer for the second phase;

two multiplier values operable to be input into the apparatus during a second phase being left hand n-k character values from the B operand, designated $\underline{B}$, and the left hand n-k characters of the N modulus, designated $\underline{N}$, respectively; and

wherein said apparatus further comprises a multiplying flush out device operative in a last multiplication phase to transfer a left hand segment of a result value remaining in the $\underline{\oplus}$ accumulation device into a result register.

19. Apparatus as in claim 18, operable to perform $\underline{\otimes}$ multiplication on polynomial based operands in a reverse mode, multiplying from MS characters to LS characters, and thereby being able to perform modular reduction without Montgomery type parasitic functions.

20. An apparatus operative in modular multiplication to anticipate a Y_0 value using first emitted values of a multiplicand, and present inputs of a B multiplier, carry out values from a $\underline{\oplus}$ accumulation device, $\underline{\oplus}$ summation values from the $\underline{\oplus}$ accumulation device, the present values from a previously computed partial result, and carry out values from a $\underline{\oplus}$ adder which $\underline{\oplus}$ summates the result from the $\underline{\oplus}$ accumulation device with the previous partial result.

21. An apparatus as in claim 20 adapted to ensure that k first emitted values from the device are zeroes, said adaptation comprising anticipation of a next in turn Y_0 character using the following quantities:

i l bit S_{out} bits of a result of l bit by l bit mod 2^l $\underline{\otimes}$ multiplication of the right-hand character of an A_i register times the B_d character of the B Stream, $A_0 \cdot B_d \bmod 2^l$;

ii a first emitting carry out character from the $\underline{\oplus}$ accumulation device, $S(CO_0)$;

iii the l bit S_{out} character from a second from the right-hand character emitting cell of the $\underline{\oplus}$ accumulation device, SO_1;

iv a next in turn character value from the S stream, S_d;

v a l bit carry out character from a Z output full adder, $S(CO_z)$;

vi a l bit J_0 value, which is a negative multiplicative inverse of a right-hand character in the N_0 modulus multiplicand register;

wherein values. $A_0 \cdot B_d \bmod 2^l$. $S(CO_0)$, SO_1, S_d are $\underline{\oplus}$ added character to character together and "on the fly" $\underline{\otimes}$ multiplied by the J_0 character to output a valid Y_0 zero-forcing anticipatory character.

22. Apparatus as in any one of claims 18 to 21, comprising at least one sensor operative to compare an output result to N. the mechanism operative to actuate a second subtractor on the output of the result register, thereby to output a modular reduced value which is limited congruent to the output result value, thereby avoiding any necessity to allot a second memory storage for a smaller result.

23. Apparatus according to any one of claims 18 to 22, wherein a value which is a $\underline{\oplus}$ summation of two multiplicands is loadable into a preload character buffer comprising at least a k character memory register whilst one of the said two multiplicands is concurrently loaded into another preload buffer.

24. Apparatus with one $\underline{\oplus}$ accumulation device, and an anticipating zero forcing mechanism, operative to perform a series of interleaved $\underline{\otimes}$ modular multiplications and squarings, and being adapted to perform concurrently the equivalent of three natural integer multiplication operations, such that a result is an exponentiation.

25. Apparatus according to any one of claims 18 to 24, wherein next in turn used multiplicands are preloaded into a preload register buffer on the fly.

26. An apparatus according to any one of claims 18 to 25, operable to $\underline{\oplus}$ sum two multiplicands into at least a k character register whilst concurrently loading one of the two multiplicands into a preload buffer.

27. Apparatus according to any one of claims 18 to 26, wherein apparatus buffers and registers are operative to be loaded with values from external memory sources and to be unloaded into an external memory source during computations, such that a maximum size of the operands is independent of sizes of said registers and said buffers.

28. Apparatus according to any one of claims 1 to 27, comprising a memory register, said memory register being typically serial –single –character -in /serial-single-character- out, parallel- at- least- k -characters- in/ parallel-at-least-k-characters-out, serial -single- character -in/ parallel –at –least –k –characters -out, and parallel- k-characters- in/ serial- single- character- out.

29. An apparatus according to any preceding claim operable to provide, during a final phase of a $\underline{\otimes}$ multiplication type iteration, at inputs of said multiplication device, a plurality of zero characters, which zero characters are operative to flush out a left hand segment of a memory of the carry save $\underline{\oplus}$ accumulator.

30. An apparatus as in any one of claims 18 to 29, operable to preload next in turn multiplicands into preload memory buffers on the fly, prior to their being required in an iteration.

31. An apparatus according to any one of claims 18 to 29, operable to preload multiplicand values into preload buffers on the fly from a central storage memory.

32. Microelectronic method for performing interleaved finite field $\underline{\otimes}$ modular multiplication of two integers A and B, so as to generate an output stream of A times B modulus N, wherein a number of characters in a modulus operand register, n, is larger than a segment length of k characters, wherein the $\underline{\otimes}$ modular multiplication is performed in a plurality of interleaved iterations, wherein at each interleaved iteration, operands are input into a $\underline{\otimes}$ multiplying device, said operands comprising any one of N, B, a previously computed partial result, S, and a k character string segment of A, a multiplicand, the segments progressing from a first string segment A_0 to a higher string segment A_{m-1}, wherein each iterative result is $\underline{\oplus}$ summated into a next temporary result, wherein at least first emitted characters of said iterations are zeroes, the method comprising the steps of:

$\underline{\oplus}$ summating into an $\underline{\oplus}$ accumulation device, in turn one or two of a plurality of multiplicand values, during each one of a plurality of phases of the iterative $\underline{\otimes}$ multiplication process, and in turn receiving as multipliers, inputs from:

a B register,

an "on the fly" anticipating value, Y_0, operable as a multiplier to force first emitting right-hand zero output characters in each iteration, and

an N, register;

generating a binary string operative to be a multiplier during a first multiplication phase and operative to be a multiplicand in a second multiplication phase;

obtaining multiplicand values suitable for switching into the $\underline{\oplus}$ accumulation device for the first multiplication phase consisting firstly of a zero value, secondly a value, A_i, which is a k character string segment of a multiplicand, A, and thirdly a value N_0, being the first emitted k characters of the modulus, N;

obtaining a temporary result value, S, resulting from a previous iteration, and $\underline{\oplus}$ summating said temporary result value, S, with a present result value emanating from the $\underline{\oplus}$ accumulation device, to generate a partial result for a next in turn iteration;

obtaining multiplicand values for a second multiplication phase comprising firstly a zero value, secondly an A_i operand, remaining in place from the first phase, and thirdly a Y_0 value having been anticipated in the first phase;

utilizing multiplier values obtained in the first phase, said values being firstly an emitted string, B_0, and multiplying said string concurrently $\underline{\otimes}$ with a second $\underline{\otimes}$ multiplier value consisting of the anticipated Y_0 string as it is simultaneously loaded character by character whilst being generated into a preload multiplicand buffer for the second phase;

obtaining two multiplier values during the second multiplication phase, said values being left hand n-k character values from the B operand, designated $\underline{B}$, and the left hand n-k characters of the N modulus, designated $\underline{N}$, respectively; and

in a last multiplication phase transferring a left hand segment of a result value remaining in the $\underline{\oplus}$ accumulation device into a result register.

33. A method according to claim 18 comprising computing $J_0=Y_0$ for l=1 by resetting both A and B to zero and setting $S_0 = 1$.

For the Applicant

Sanford T. Colb & Co.
C: 38107

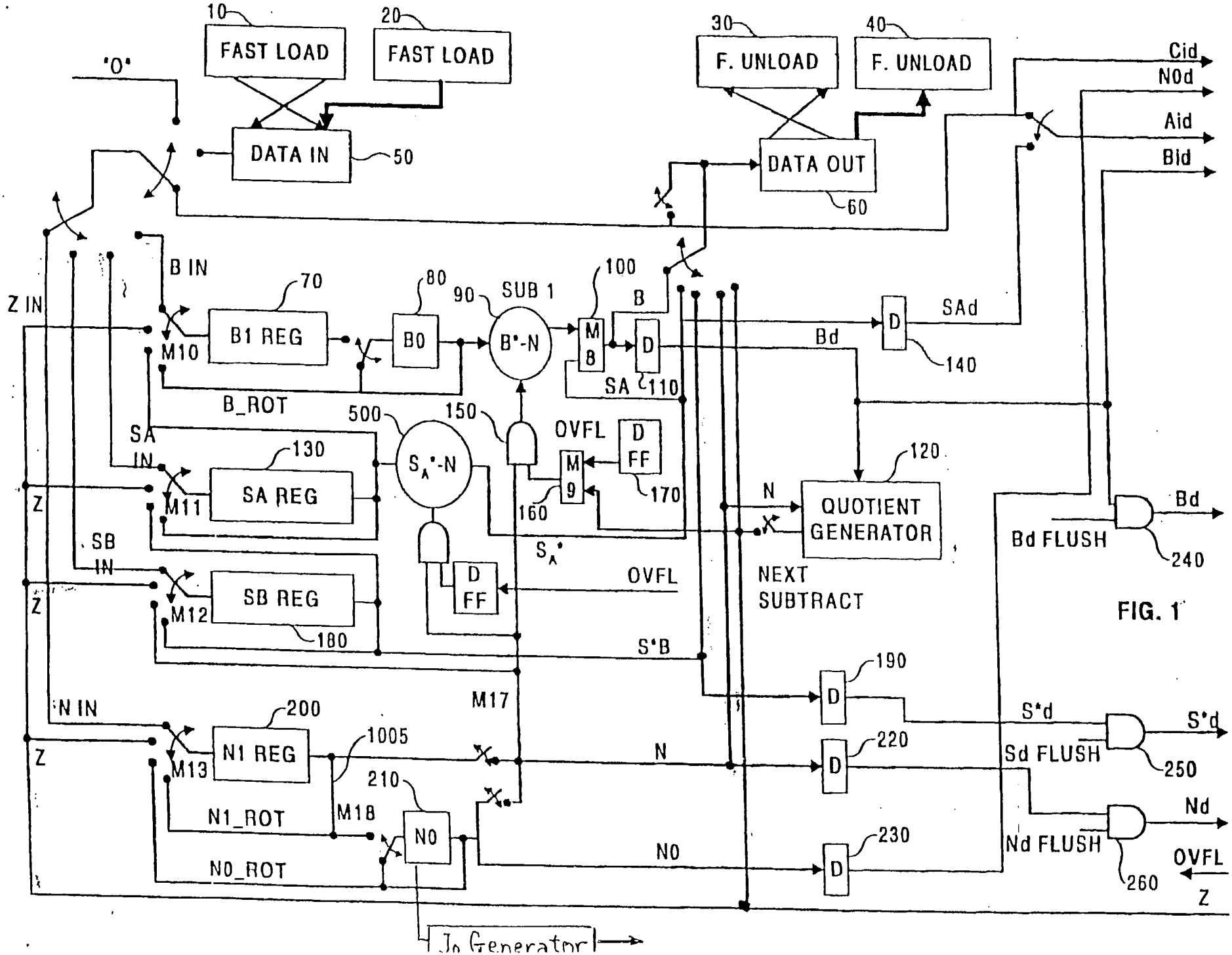

10
FAST LOAD
20
FAST LOAD
"0"
DATA IN
50
30
F. UNLOAD
40
F. UNLOAD
DATA OUT
60
Cid
NOd
Aid
Bid
B IN
Z IN
70
B1 REG
M10
80
B0
90
SUB 1
B*-N
100
M 8
B
D
SAd
Bd
140
SA
110
B_ROT
500
150
OVFL
D FF
M 9
160
170
SA IN
130
SA REG
M11
Z
S_A*-N
120
N
QUOTIENT GENERATOR
Bd
Bd FLUSH
240
SB IN
SB REG
M12
180
Z
D FF
OVFL
S_A*
NEXT SUBTRACT
S*B
190
M17
N IN
200
N1 REG
M13
1005
Z
N
220
S*d
Sd FLUSH
250
S*d
210
N1_ROT
M18
N0
N0
230
Nd
Nd FLUSH
260
OVFL
N0_ROT
Z
Jn Generator

FIG. 1

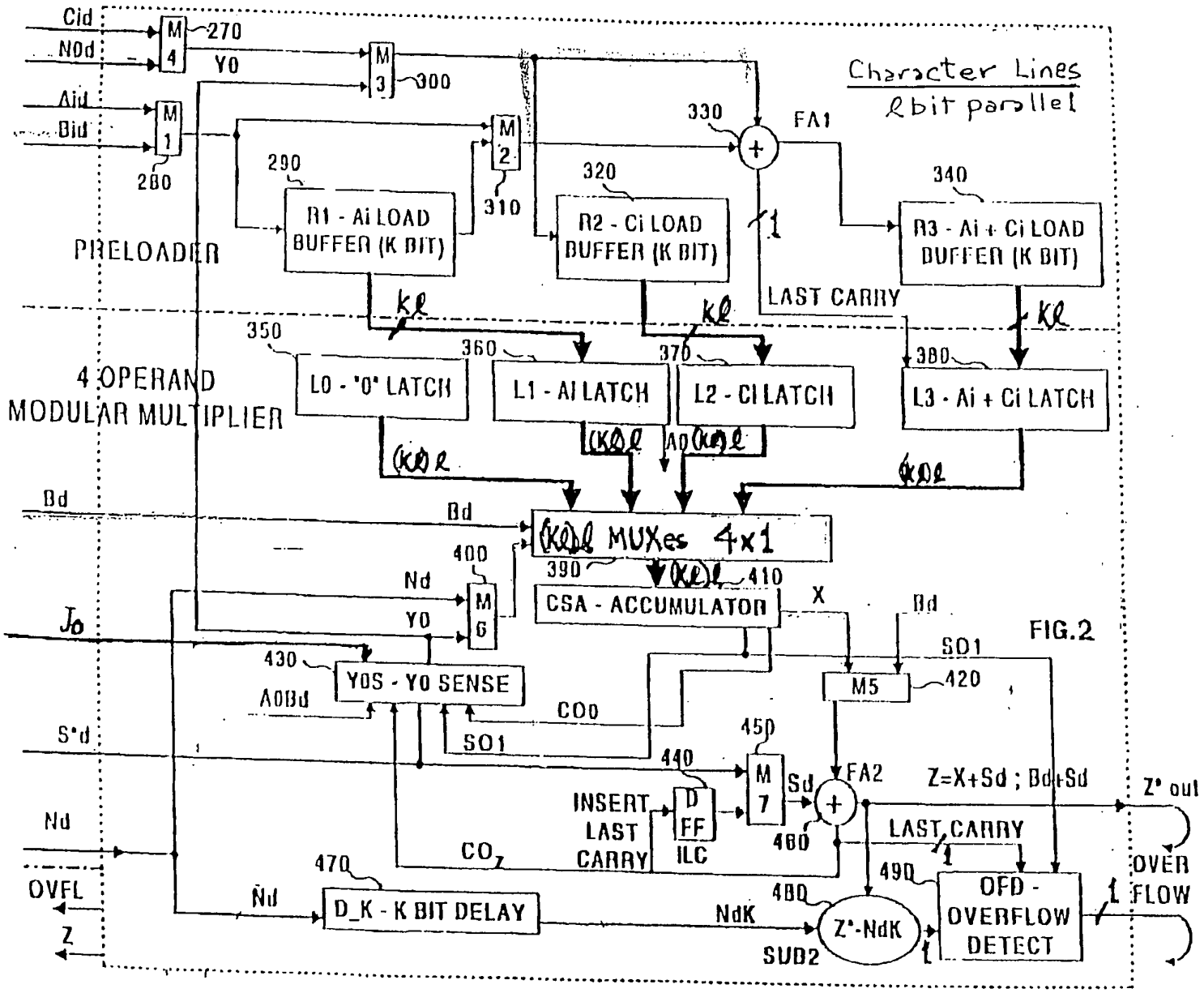

Cid
NOd
Aid
Bid
M 4
M 3
M 1
M 2
270
300
280
310
Y0
290
320
330
FA1
340
Character Lines
ℓ bit parallel
R1 - Ai LOAD BUFFER (K BIT)
R2 - Ci LOAD BUFFER (K BIT)
R3 - Ai + Ci LOAD BUFFER (K BIT)
PRELOADER
LAST CARRY
4 OPERAND MODULAR MULTIPLIER
350
360
370
380
L0 - "0" LATCH
L1 - Ai LATCH
L2 - Ci LATCH
L3 - Ai + Ci LATCH
Bd
(K)ℓ MUXes 4x1
400
390
410
Nd
M 6
CSA - ACCUMULATOR
X
Jo
Y0
FIG.2
430
Y0S - Y0 SENSE
M5
420
A0Bd
CO0
SO1
S*d
450
440
M 7
Sd
FA2
Z=X+Sd ; Bd+Sd
Z* out
INSERT LAST CARRY
D FF
ILC
460
LAST CARRY
Nd
CO_Z
OVER FLOW
OVFL
470
490
Nd
D_K - K BIT DELAY
480
NdK
Z*-NdK
OFD - OVERFLOW DETECT
Z
SUB2

Fig. 2

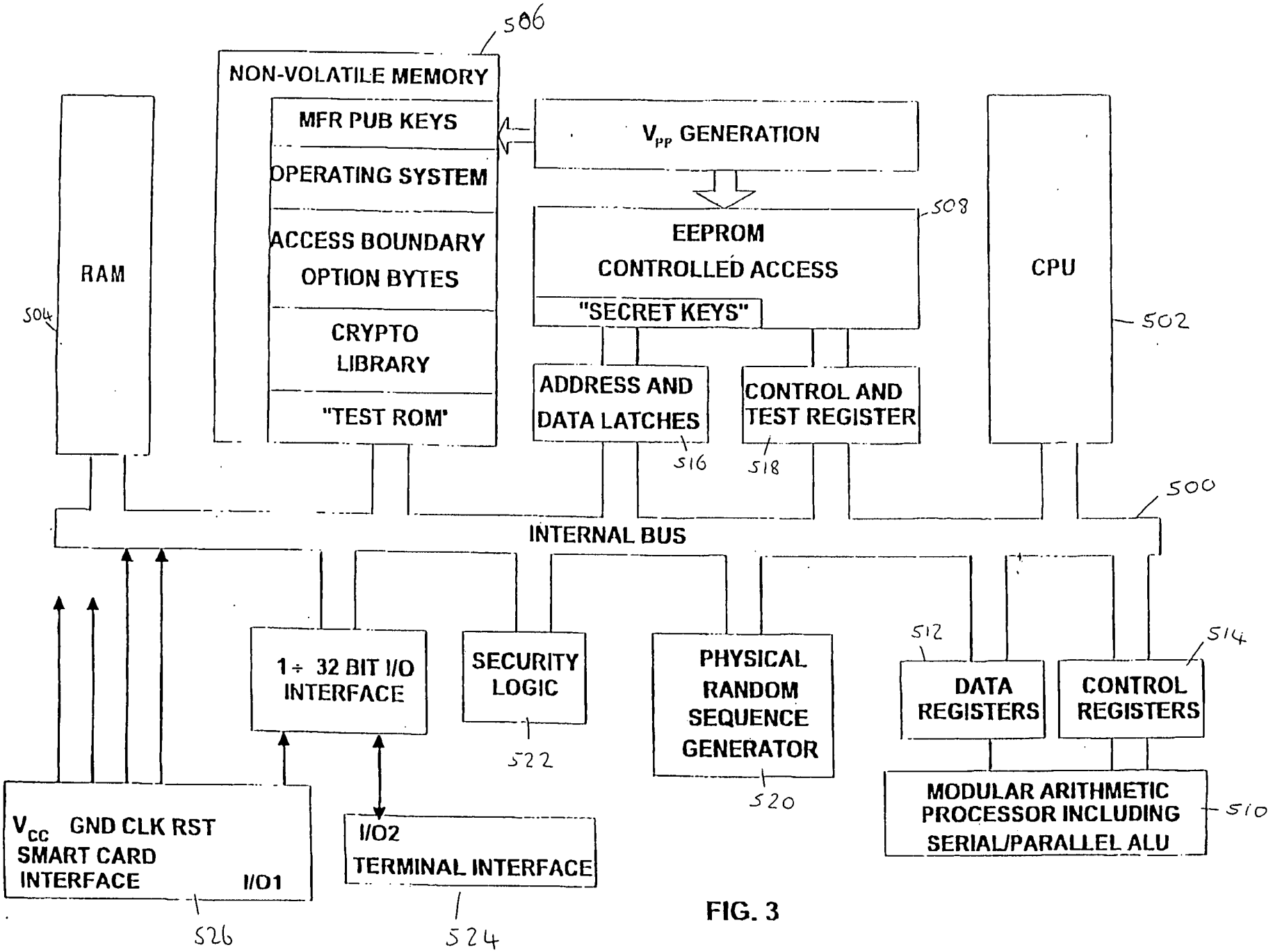

506
NON-VOLATILE MEMORY
MFR PUB KEYS
OPERATING SYSTEM
ACCESS BOUNDARY
OPTION BYTES
CRYPTO
LIBRARY
"TEST ROM"
V_{PP} GENERATION
508
EEPROM
CONTROLLED ACCESS
"SECRET KEYS"
RAM
504
CPU
502
ADDRESS AND
DATA LATCHES
516
CONTROL AND
TEST REGISTER
518
500
INTERNAL BUS
1÷ 32 BIT I/O
INTERFACE
SECURITY
LOGIC
522
PHYSICAL
RANDOM
SEQUENCE
GENERATOR
520
512
DATA
REGISTERS
514
CONTROL
REGISTERS
MODULAR ARITHMETIC
PROCESSOR INCLUDING
SERIAL/PARALLEL ALU
510
V_{CC} GND CLK RST
SMART CARD
INTERFACE
I/O1
526
I/O2
TERMINAL INTERFACE
524

FIG. 3

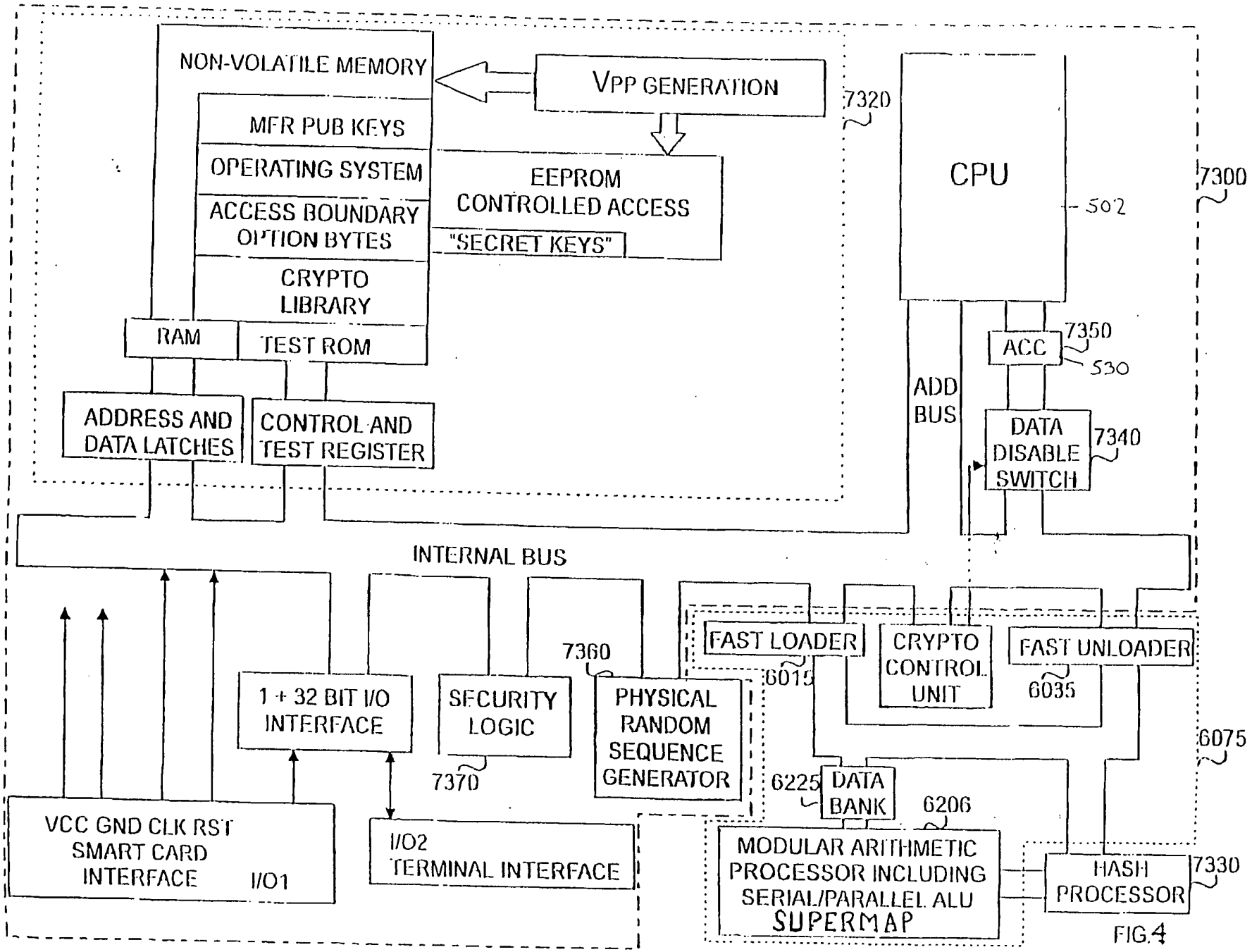

NON-VOLATILE MEMORY
VPP GENERATION
7320
MFR PUB KEYS
OPERATING SYSTEM
EEPROM CONTROLLED ACCESS
ACCESS BOUNDARY OPTION BYTES
"SECRET KEYS"
CRYPTO LIBRARY
CPU
502
7300
RAM
TEST ROM
7350
ACC
530
ADD BUS
ADDRESS AND DATA LATCHES
CONTROL AND TEST REGISTER
DATA DISABLE SWITCH
7340
INTERNAL BUS
FAST LOADER
6015
CRYPTO CONTROL UNIT
FAST UNLOADER
6035
7360
1 + 32 BIT I/O INTERFACE
SECURITY LOGIC
PHYSICAL RANDOM SEQUENCE GENERATOR
6075
7370
6225
DATA BANK
6206
VCC GND CLK RST SMART CARD INTERFACE
I/O1
I/O2 TERMINAL INTERFACE
MODULAR ARITHMETIC PROCESSOR INCLUDING SERIAL/PARALLEL ALU SUPERMAP
HASH PROCESSOR
7330

FIG.4

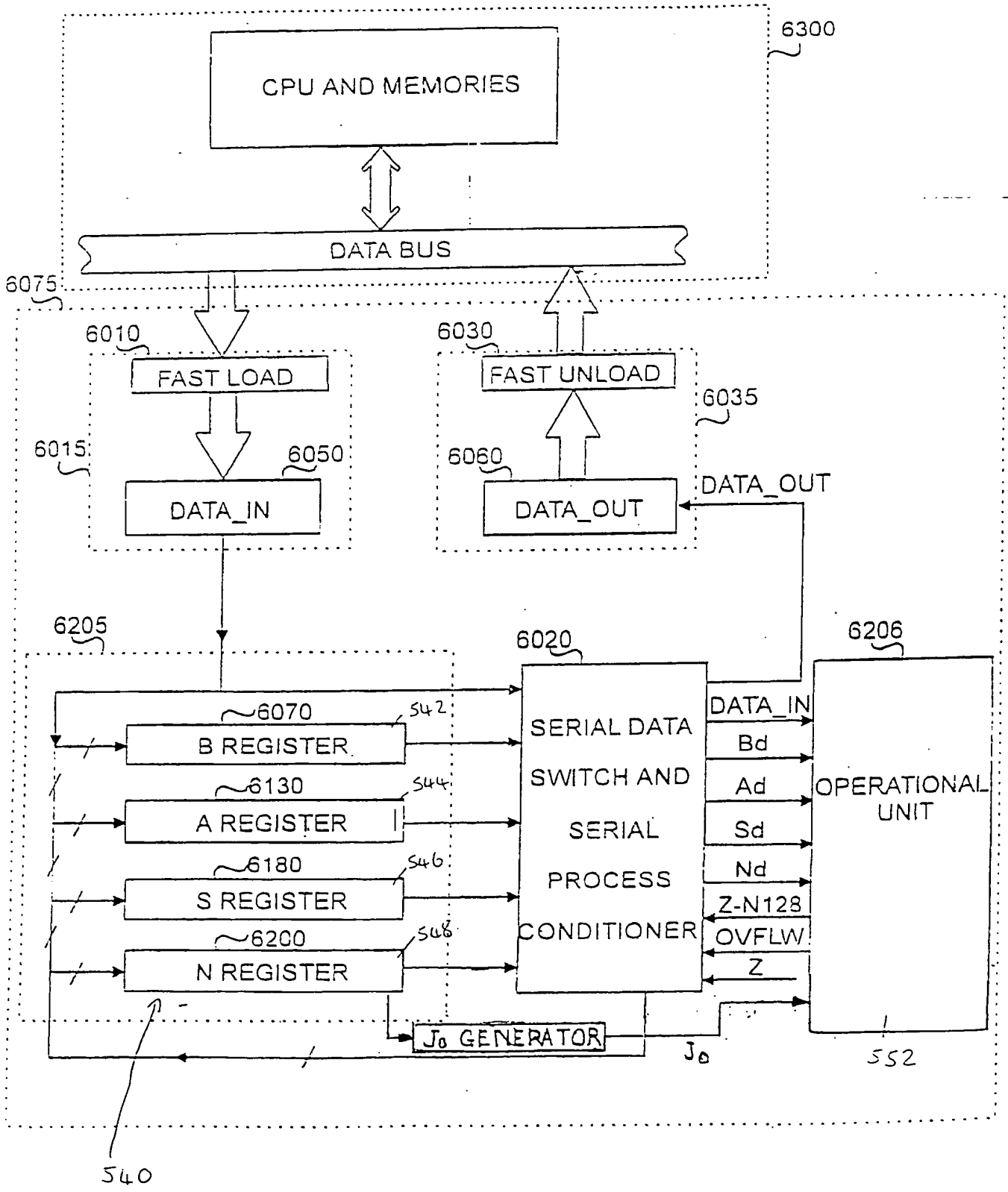
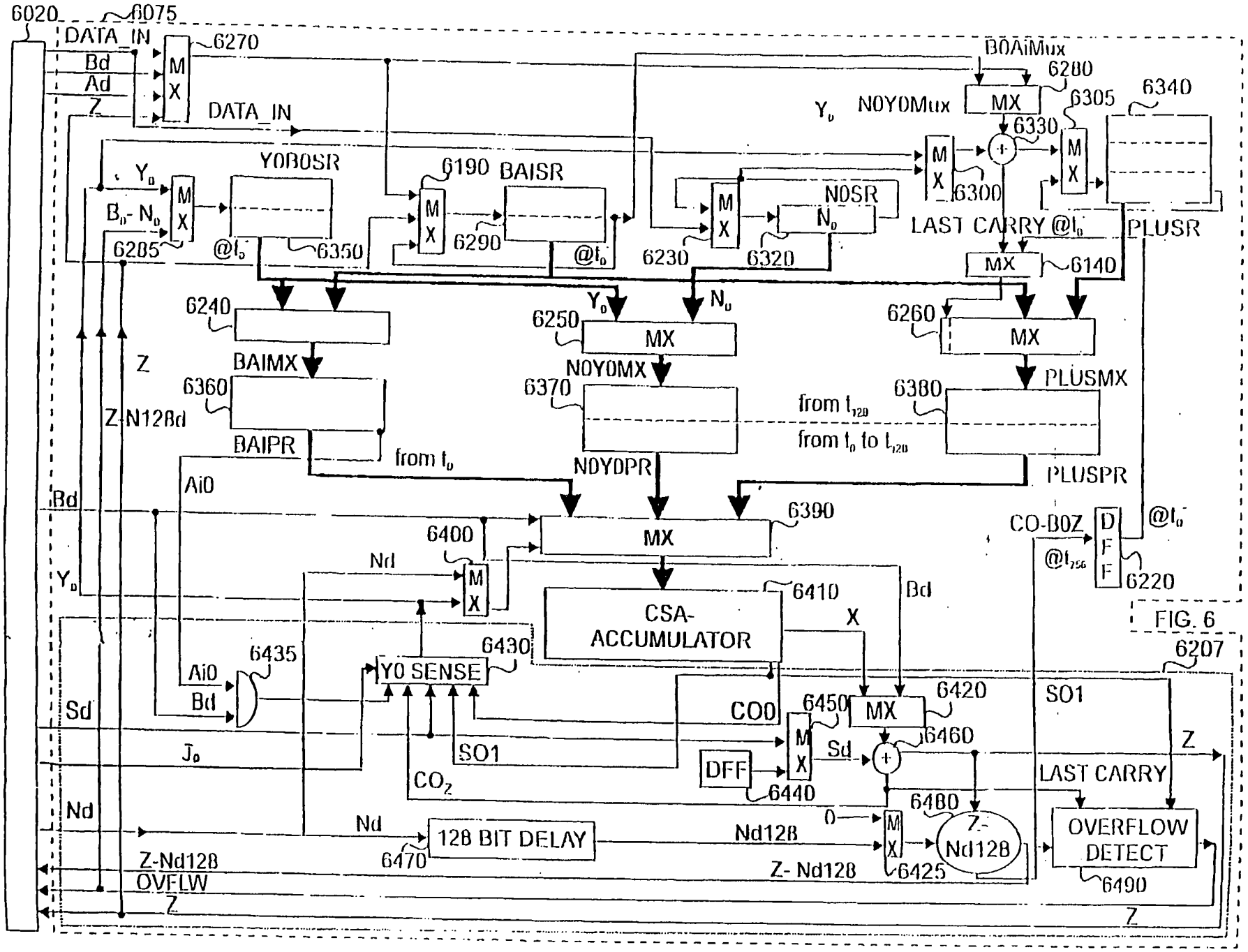

6300
CPU AND MEMORIES
DATA BUS
6075
6010
FAST LOAD
6015
6050
DATA_IN
6030
FAST UNLOAD
6035
6060
DATA_OUT
DATA_OUT
6205
6020
6206
6070
B REGISTER
542
6130
A REGISTER
544
6180
S REGISTER
546
6200
N REGISTER
548
SERIAL DATA
SWITCH AND
SERIAL
PROCESS
CONDITIONER
DATA_IN
Bd
Ad
Sd
Nd
Z-N128
OVFLW
Z
OPERATIONAL
UNIT
J₀ GENERATOR
J₀
552
540

FIG. 5


6020
6075
DATA_IN
Bd
Ad
Z
MX
6270
DATA_IN
B0AiMux
6280
MX
Y_0
N0Y0Mux
6305
6340
6330
Y0B0SR
Y_0
B_0- N_0
6285
@t_0
6350
6190 BAISR
6290
@t_0
N0SR
N_0
6230
6320
6300
LAST CARRY @t_0
PLUSR
MX
6140
6240
BAIMX
Z
Z-N128d
6360
BAIPR
from t_0
Y_0
N_0
6250
MX
N0Y0MX
6370
N0Y0PR
6260
MX
PLUSMX
6380
from t_{128}
from t_0 to t_{128}
PLUSPR
Bd
Ai0
6400
Nd
Y_0
MX
6390
MX
6410
CSA-
ACCUMULATOR
Bd
X
CO-B0Z
@t_{256}
D
F
F
@t_0
6220
FIG. 6
6207
6435
Ai0
Bd
Sd
6430
Y0 SENSE
CO0
6450
MX
6420
SO1
J_0
SO1
CO_2
DFF
6440
Sd
6460
Z
LAST CARRY
Nd
Nd
6470
128 BIT DELAY
Nd128
0
6480
Z-
Nd128
OVERFLOW
DETECT
6490
Z-Nd128
OVFLW
Z
Z- Nd128
6425
Z

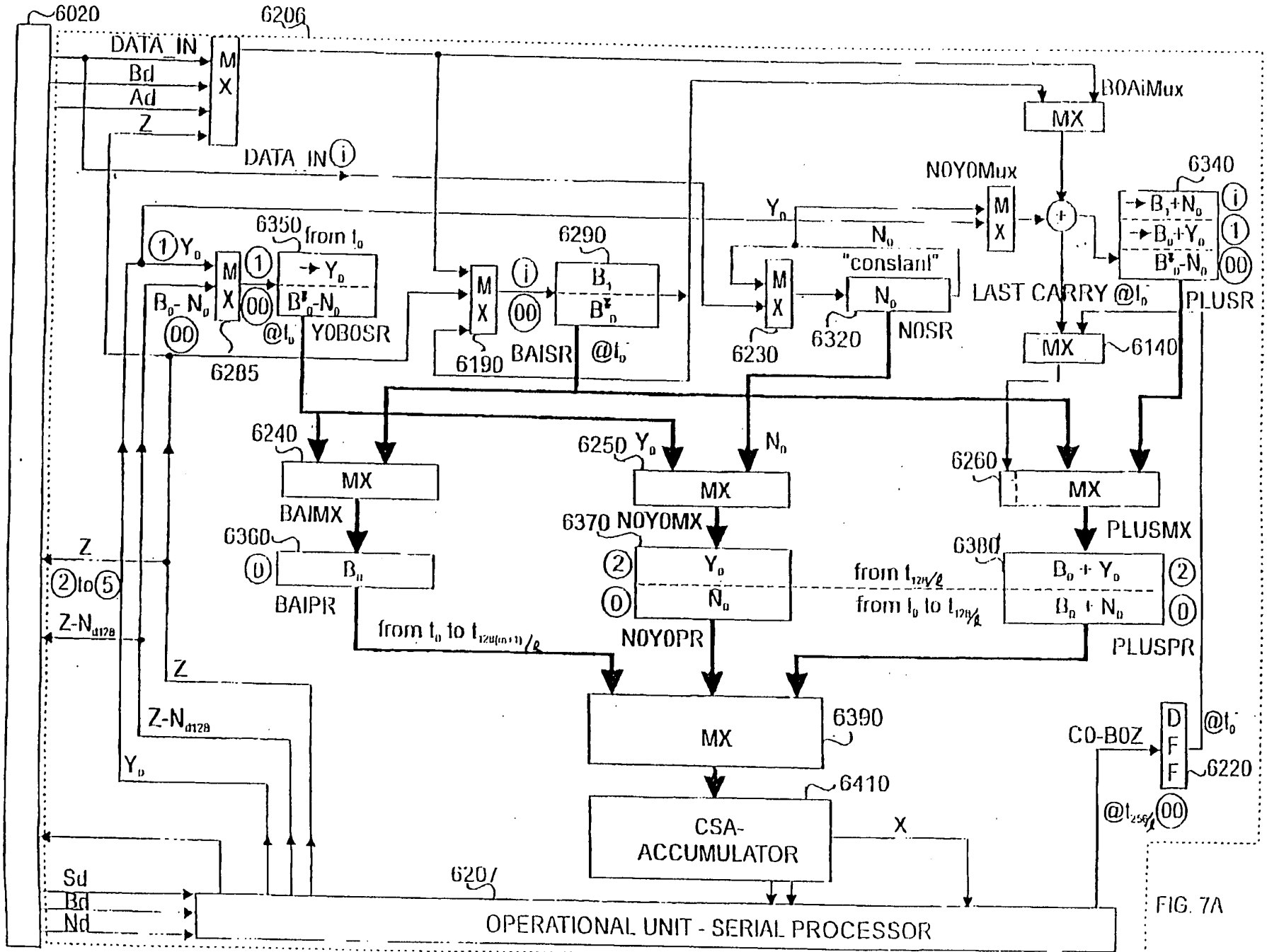

6020
6206
DATA IN
Bd
Ad
Z
MX
DATA IN (i)
B0AiMux
NOY0Mux
6340
B_1+N_0
B_0+Y_0
$B^*_0-N_0$
LAST CARRY @t_0
PLUSR
Y_0
N_0
"constant"
6350
from t_0
B_0-N_0
@t_0
Y0B0SR
6285
6190
BAISR
@t_0
6290
B_1
B^*_0
6230
6320
N0SR
6140
6240
BAIMX
6250 Y_0
N_0
6260
PLUSMX
6360
B_0
BAIPR
6370 NOY0MX
Y_0
N_0
from t_{128}/ℓ
from t_0 to t_{128}/ℓ
6380
B_0+Y_0
B_0+N_0
PLUSPR
Z
2 to 5
Z-N_{d128}
from t_0 to $t_{128(n+1)}$/ℓ
NOY0PR
6390
C0-B0Z
@t_0
6220
@t_{256}/ℓ
6410
CSA-
ACCUMULATOR
X
Sd
Bd
Nd
6207
OPERATIONAL UNIT - SERIAL PROCESSOR

FIG. 7A

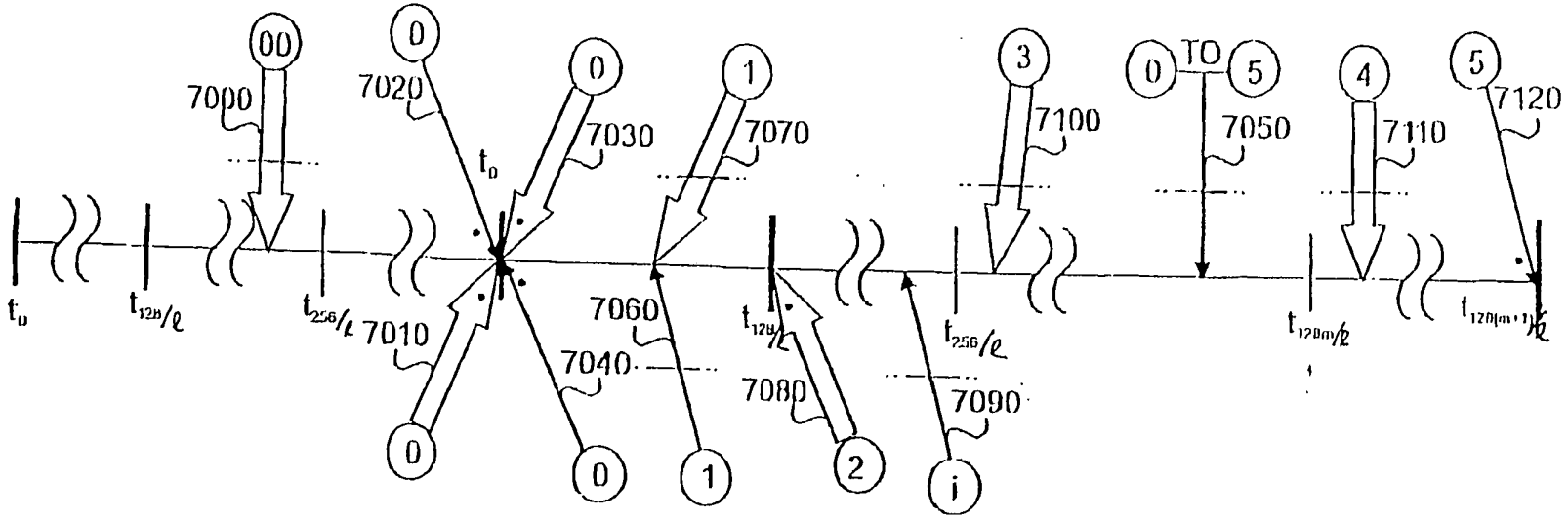

00
0
0
1
3
0
TO
5
4
5
7000
7020
7030
7070
7100
7050
7110
7120
t_0
t_{128}/ℓ
t_{256}/ℓ
7010
7060
7040
t_{128}/ℓ
7080
t_{256}/ℓ
7090
t_{128m}/ℓ
$t_{128(m+1)}/\ell$
0
0
1
2
i

FIG. 7B

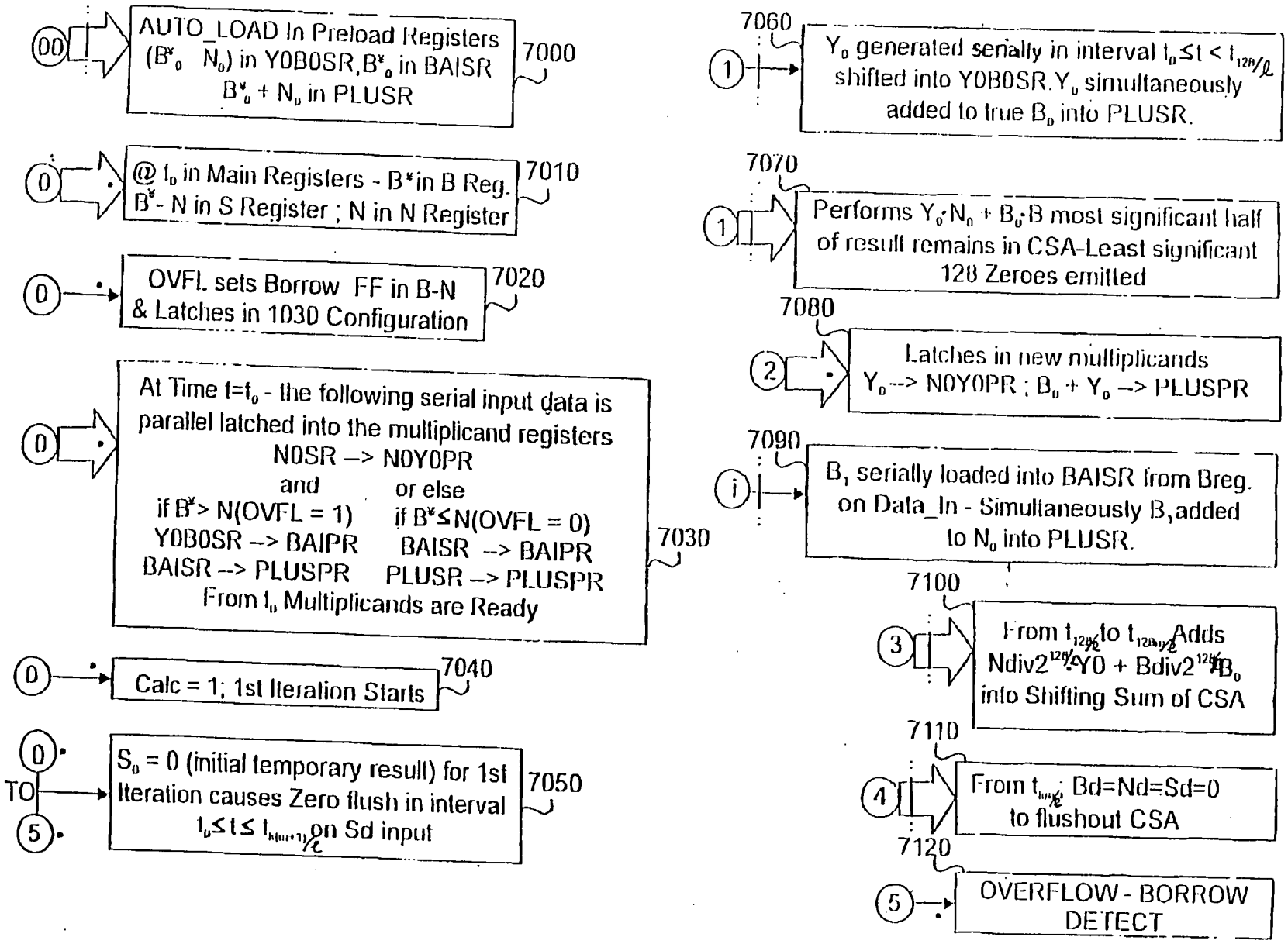

00
AUTO_LOAD In Preload Registers
(B^*_0 N_0) in Y0B0SR, B^*_0 in BAISR
$B^*_0 + N_0$ in PLUSR
7000
0
@ t_0 in Main Registers - B^* in B Reg.
B^* - N in S Register ; N in N Register
7010
0
OVFL sets Borrow FF in B-N
& Latches in 1030 Configuration
7020
0
At Time $t=t_0$ - the following serial input data is parallel latched into the multiplicand registers
N0SR --> N0Y0PR
and
or else
if $B^* > N$ (OVFL = 1)
if $B^* \leq N$ (OVFL = 0)
Y0B0SR --> BAIPR
BAISR --> BAIPR
BAISR --> PLUSPR
PLUSR --> PLUSPR
From t_0 Multiplicands are Ready
7030
0
Calc = 1; 1st Iteration Starts
7040
0
TO
5
$S_0 = 0$ (initial temporary result) for 1st Iteration causes Zero flush in interval $t_0 \leq t \leq t_{k(m+1)/\ell}$ on Sd input
7050
7060
1
Y_0 generated serially in interval $t_0 \leq t < t_{128/\ell}$ shifted into Y0B0SR. Y_0 simultaneously added to true B_0 into PLUSR.
7070
1
Performs $Y_0 \cdot N_0 + B_0 \cdot B$ most significant half of result remains in CSA-Least significant
128 Zeroes emitted
7080
2
Latches in new multiplicands
Y_0 --> N0Y0PR ; $B_0 + Y_0$ --> PLUSPR
7090
i
B_1 serially loaded into BAISR from Breg. on Data_In - Simultaneously B_1 added to N_0 into PLUSR.
7100
3
From $t_{128/\ell}$ to $t_{128(m+2)/\ell}$ Adds
$Ndiv2^{128/\ell} \cdot Y0 + Bdiv2^{128/\ell} \cdot B_0$
into Shifting Sum of CSA
7110
4
From $t_{k(m+1)/\ell}$ Bd=Nd=Sd=0
to flushout CSA
7120
5
OVERFLOW - BORROW
DETECT

FIG. 7C

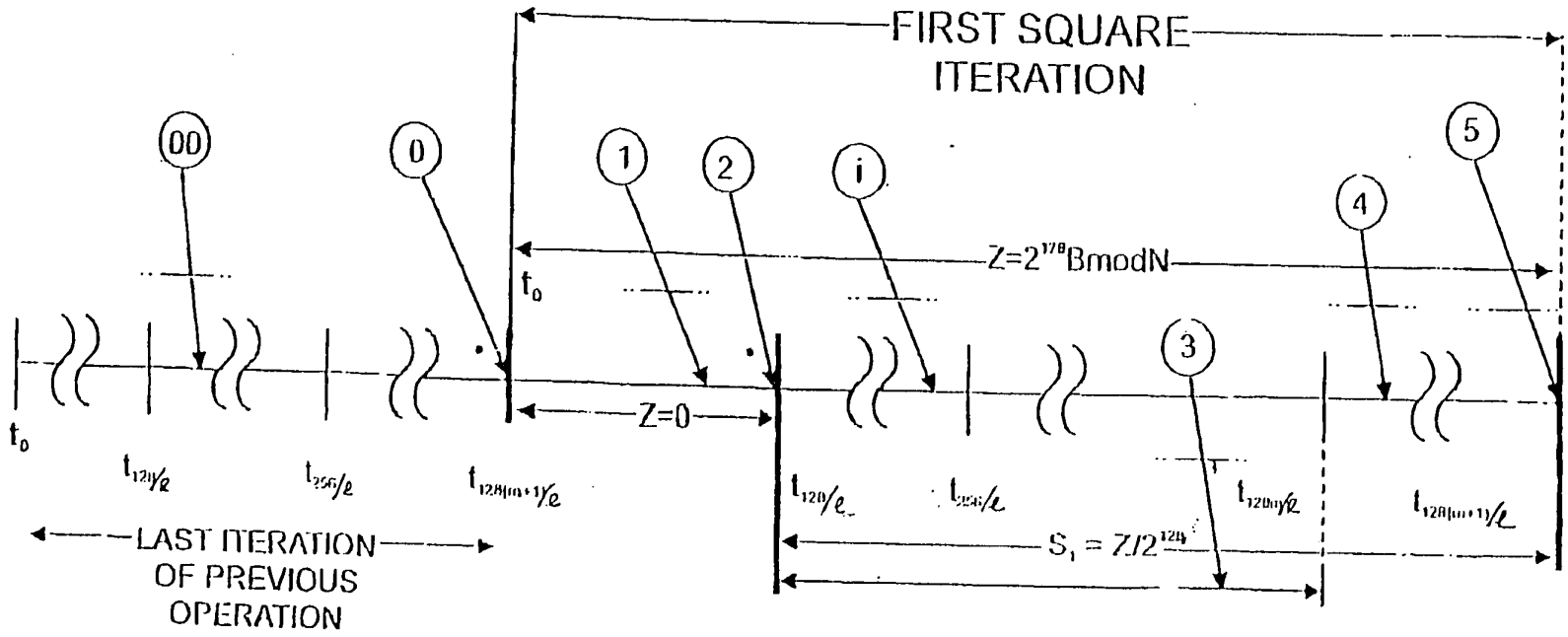

FIRST SQUARE
ITERATION
00
0
1
2
i
3
4
5
$Z=2^{128}BmodN$
t_0
$Z=0$
$t_{128/e}$
$t_{256/e}$
$t_{128(m+1)/e}$
$S_1 = Z/2^{128}$
LAST ITERATION
OF PREVIOUS
OPERATION

FIG. 7D

	N_o	N_o^{-1}	$-(N_o^{-1})$
1	0001	0001	1111
3	0011	1011	0101
5	0101	1101	0011
7	0111	0111	1001
9	1001	1001	0111
11	1011	1011	0101
13	1101	0101	1011
15	1111	1111	0001

$\delta = 1$

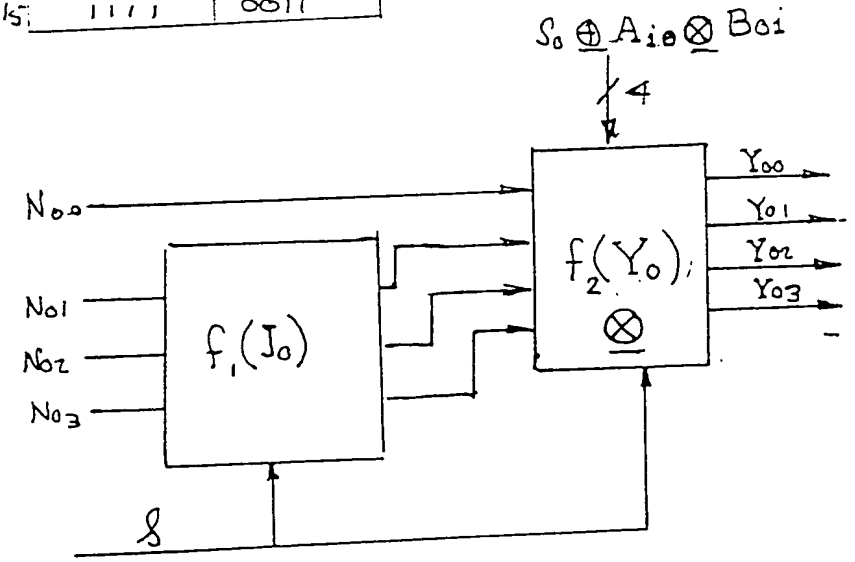
$GF(p)$

	N_o	$-N_o^{-1} = N_o^{-1}$
1	0001	0001
3	0011	1111
5	0101	0101
7	0111	1011
9	1001	1001
11	1011	0111
13	1101	1101
15	1111	0011

$\delta = 0$

No Carry

$GF(2^9)$



$l = 4$

Fig. 8